ORACLE

JD EDWARDS WORLD

JD Edwards World Programmer's Guide

Version A7.3

Copyright Notice

Copyright © 2012, Oracle. All rights reserved.

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and / or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Subject to patent protection under one or more of the following U.S. patents: 5,781,908; 5,828,376; 5,950,010; 5,960,204; 5,987,497; 5,995,972; 5,987,497; and 6,223,345. Other patents pending.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal
Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

Contains GNU libgmp library; Copyright 1991 Free Software Foundation, Inc. This library is free software which
can be modified and redistributed under the terms of the GNU Library General Public License.

Includes Adobe PDF Library, Copyright 1993-2001 Adobe Systems, Inc. and DL Interface, Copyright 1999-2008 Datalogics Inc. All rights reserved. Adobe is a trademark of Adobe Systems Incorporated.

Portions of this program contain information proprietary to Microsoft Corporation. Copyright 1985-1999 Microsoft Corporation.

Portions of this program contain information proprietary to Tenberry Software, Inc. Copyright 1992-1995 Tenberry Software, Inc.

Portions of this program contain information proprietary to Premia Corporation. Copyright 1993 Premia Corporation.

This product includes code licensed from RSA Data Security. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software written by Tim Hudson (tjh@cryptsoft.com). All rights reserved.

This product includes the Sentry Spelling-Checker Engine, Copyright 1993 Wintertree Software Inc. All rights reserved.

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 JD Edwards World products and the following disclaimers are provided.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

Overview

This publication provides detailed technical information for the A7.3 software release from J.D. Edwards. It is written for programmers that integrate other software with J.D. Edwards software, or customize J.D. Edwards software for their particular needs. It does not provide an understanding of the "big picture," or how programs work together within the J.D. Edwards system, but it has details of specific interest to the programmer.

What's in this Publication

This publication includes database and system changes for the following product groups:

General and Financial systems

Distribution and Manufacturing systems

Human Resources and Payroll systems

Architecture, Engineering, and Construction systems

Database changes include such items as fields added or changed, new or obsolete files, and new or changed logical files.

System changes describe various enhancements, such as new servers, programs, and files. The descriptions provide helpful information you should be aware of as you integrate or customize J.D. Edwards software.

In addition, this publication includes information about:

Re-engineering modules and conversion tools

Performance considerations

National language support

Other Publications

Use this publication with the technical upgrade notes, which are provided by J.D. Edwards in two ways:

The *Technical Upgrade Notes Guide (Stand-Alone Version - JDETUN)* with accompanying tape

The Work With Tech Upgrade Notes selection on the Software Upgrade Menu (A97IBM)

NOTE: For more information about this selection, refer to the *Upgrade Reference Guide.*

Other helpful publications include:

Reinstallation Workbook

Upgrade Reference Guide

If you have suggestions for items to include or not include in this document, call your J.D. Edwards Worldwide Customer Support representative.

General and Financial Systems

This section provides the following:

Database changes for financial systems and electronic commerce

PPAT (J.D. Edwards E-Mail) system design changes

Database Changes

This chapter lists database changes for financial systems and electronic commerce.

Financial Systems

Fields Added

File	**Fields Added**
F0312	VR01 - Reference AR - Reason Code
F092181	ALA - Alternate Account AAD - Alternate Account Description

New Logical Files

File	Key Fields
F01131LA	SERK, AN8, DTI
F01131LB	PA8, DTI, MBDS
F01131LC	PA8, DTI (descending)
F01131LD	AN8, DSS5, DTI
F01131LE	AN8, DTI, MBDS
F01131LF	AN8, DTI (descending)
F01131LG	AN8, DSS5
F01131LH	PA8, DSS5
F01132LA	SERK, LIN
F01133LA	AN8, DTI, MBDS
F01133LB	AN8, MBDS, DTI (descending)
F01133LC	AN8, MBDS, DSS5, DTI
F01133LD	SERK, STSM, DTI
F01133LE	AN8, MBDS, STSM, DTI (descending)
F01133LF	AN8, MBDS, DTI
F01133LG	AN8, MBDS, STSM, DTI
F01133LH	AN8, MBDS, DSS5
F01134LA	USER, DSS5
F01134LB	AN8, USER
F01134LC	UGRP, DSS5
F01134LD	AN8, UGRP
F01134LE	UGRP, AN8
F0311LY	VR01, AN8
F092181	N001, ALA, MCU, OBJ, SUB

Logical File Changes

File	New Key Fields Underlined
F0911LJ	<u>ICUT</u>, <u>ICU</u>, CO, FY, PN

New Files

File	File Name	File Contents	Prefix
F01131	PPAT Message Header	PPAT Message Header	ZZ

F01132	PPAT Message Text	PPAT Message Text	CY
F01133	PPAT Message Distribution	PPAT List Distribution Info	ZT
F01134	Bulletin Board Enrollment	Information about who is subscribed to bulletin boards	ZW
F01136	Incoming Mail Filters	Information to support PPAT mail filter feature	ZP
T011121	Outbound Internet Workfile	Outbound Internet Messages	CT

Obsolete Files

File	Description
F0113	Message Log Ledger File - Obsolete Physical (replaced by F01131, F01132, F01133)
F0113LA	Obsolete logical over F0113
F0113LB	Obsolete logical over F0113
F0113LE	Obsolete logical over F0113

Electronic Commerce

Fields Added

File	Fields Added
F4706	AN8 - Address Number

New Files

File	File Name	File Contents	Prefix
F470161	EDI P.O. Additional Header - Outbound	This file contains additional tax and terms data	Z3
F470171	EDI P.O. Additional Detail - Outbound	This file contains additional tax and terms data	Z4
F471061	EDI Response to RFQ Additional Header - Outbound	This file contains additional tax and terms data	Z3
F471071	EDI Response to RFQ Additional Detail - Outbound	This file contains additional tax and terms data	Z4
F471861	EDI Product Transfer and Resale Report Additional Header - Outbound	This file contains additional tax and terms data	S3
F471871	EDI Product Transfer and Resale Report Additional Detail - Outbound	This file contains additional tax and terms data	S4

PPAT (J.D. Edwards E-Mail) System Design

This chapter describes changes made to the J.D. Edwards e-mail (PPAT) system.

File Normalization

As of release A7.3, the Message Log Ledger File has been broken up into three different files as part of a significant enhancement to the PPAT system. We also added two new files to support new features in the system. Following is a brief description of the type of information contained in each file. The final section is an explanation on how to use the called program X00PPAT1 to send automatic e-mail messages from within other programs.

NOTE: The PPAT system is now capable of sending e-mail via the Internet. However, that facility is not documented here. While it is a fully-working function that we are currently using internally, there are limitations to its use at this time and we are working on further enhancements that you may want to receive if you are would like to enable this capability. If you are interested in this feature, please contact the Denver Support Line and they will establish a development contact for you to explain how to enable the connection.

F01131: PPAT Message Header

This file contains the information from the header portion of a PPAT message such as Sender, Recipient, and Mailbox ID. This file is uniquely keyed on the following field:

Key Value Serial Number	(ZZSERK)

This key is simply a system-assigned next number from Next Number System Table 02, Index 01. The value itself is of no interest to a user of the system.

F01132: PPAT Message Text

This file contains the detail text of a PPAT message. It is uniquely keyed on the following fields:

Key Value Serial Number	(CYSERK)
Command Flag	(CYCMDF)
Line Number	(CYLIN)

The command flag is a one-character field that is used to indicate that the information on the record is an AS/400 command string as opposed to viewable message text. For more information on this feature, see the description of the X00PPAT1 module later in this document.

F01133: PPAT Message Distribution

As part of the system redesign, we are no longer creating separate copies of messages sent to multiple people through a distribution list. We are now storing only one copy of the message in the F01131 and F01132 files. This file is then used to store information about the list of recipients and the status of the message in each of their mailboxes. This file is uniquely keyed on the following fields:

Key Value Serial Number	(ZTSERK)
Recipient Address Number	(ZTAN8)

F01134: Bulletin Board Enrollment

A new system feature in A7.3 is the ability to establish bulletin boards that users may subscribe to if they are interested in viewing the contents. This file contains information about who has subscribed to each bulletin board. This file is uniquely keyed on the following fields:

User ID	(ZWUSER)
User Class/Group	(ZWUGRP)
Bulletin Board Address Number	(ZWAN8)

F01136: Incoming Mail Filters

Another new system feature is the ability to separate or "pre-sort" incoming messages into three different mailboxes (Priority Mail, Personal In Box, or Junk Mail) depending on who the sender is or which distribution list was used to send the message. This file contains information about which filters have been established by each system user. It is uniquely keyed on the following fields:

Sending User or Distribution List Address	(ZPPA8)
Recipient Address	(ZPAN8)

X00PPAT1: E-Mail Message Server

This program can be called from within any other program and will send an automatic e-mail message to a recipient. For example, in the standard J.D. Edwards software, we use this module to send messages from our Purchase Requisition Entry program to the approver of each requisition. The message alerts the approver that there is a requisition awaiting their approval.

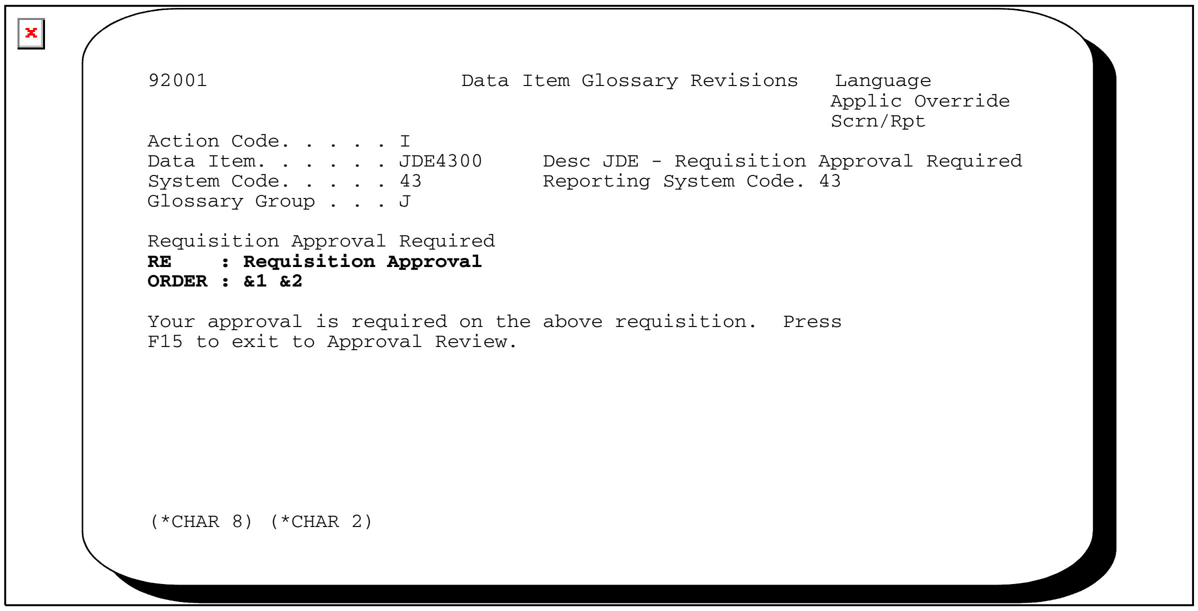
An additional optional feature of this module is the ability to turn the message into an "active message". When F15 is pressed on the message text viewing screen on an active message, the user will be automatically exited to a program that will help them carry out the requested action. In the above purchasing example, we use F15 to send the approver directly to the Purchase Requisition Approval program with the required requisition record preloaded on the screen for their review. Anything you can do from an AS/400 command line can also be done with an active message.

Creating an Automatic Message

▶ To Create an Automatic Message

Following are the three mandatory steps needed to create an automatic message. The fourth step describes the optional second call to X00PPAT1 that is necessary if you want to make your message an active message.

1. A prerequisite to the use of this program is to create a message "template" in the J.D. Edwards data dictionary. A template contains the static text of the message you want to send from your program along with place holders for variable text items you want to include at run time. As an example, the following screen represents the dictionary template we use in the purchasing example described above.



```
92001                      Data Item Glossary Revisions   Language
                                                         Applic Override
                                                         Scrn/Rpt

Action Code. . . . . I
Data Item. . . . . . JDE4300      Desc JDE - Requisition Approval Required
System Code. . . . . 43           Reporting System Code. 43
Glossary Group . . . J

Requisition Approval Required
RE    : Requisition Approval
ORDER : &1 &2

Your approval is required on the above requisition.  Press
F15 to exit to Approval Review.






(*CHAR 8) (*CHAR 2)
```

The text in bold type is the actual message the user will see. Notice the characters '&1' and '&2'. These are the "placeholders" for variable text that you want substituted in at runtime. In this case, we want to include the specific purchase requisition document number and document type in each message. The information at the very bottom of the screen is the definition of the data items that are going to be substituted in place of the '&' items. In this case, Document Number is an eight-byte character field that will be substituted for '&1' and Document Number is a two-byte character field that will be substituted for '&2'. The definitions must be entered on the last line of the screen exactly as shown.

NOTE: The data item name must start with the characters 'JDE' and the item must be created in glossary group 'J'.

2. Once the data dictionary item is in place, you must make it available to X00PPAT1 by running a batch job that will write the template from the

data dictionary to an IBM system message file. The job to be run is J98DDMSGF. It is option 2 on menu G9642.

3. Here is a sample of the RPG code needed to call X00PPAT1 to send a message:

```
0136.00     C               CALL 'X00PPAT1'
0137.00     C               ----  --------
0138.00     C               PARM        DSPARM
0139.00     C               PARM        DSDATA
0140.00     C*              PARM        PSVERS 10
0141.00     C*              PARM        PSERR   4
```

Explanation of parameters:

I/O	Parameter	Description
Input	DSPARM	A data structure containing the destination address, J.D. Edwards message template ID, and serial number (optional - see the Command String definition section below for more information on this parameter). This data structure is made available by including module I00PPAT by using the /COPY command.
Input	DSDATA	Variable data to be substituted into the template at runtime. DSDATA should be a concatenated string formatted to the length of the variables defined in the template. In our example, if the message is alerting an approver to review document number '02198003', document type 'OR', then this parameter would be formatted as '02198003OR'. The program will break the string up according to the field definitions on the template.
Input	PSVERS	The DREAM Writer version of X00PPAT1 that you want to call. This is used if you want to control the mailbox that the message will be sent to (Mailbox ID is a processing option on X00PPAT1).
Output	DSPARM	The only field in this data structure that is relevant for output is the Serial Number for the PPAT message that was just created. This value is needed if you are going to turn this message into an active message as described below.
Output	PSERR	Error message ID.

NOTE: Additional details about the parameters and how to use them are included as comments in the source code for X00PPAT1.

► Second Call to X00PPAT1 to Create an Active Message (Optional)

This step is actually a repeat of the above three steps with some minor changes. As mentioned earlier, you can do anything with an active message that you can do from an AS/400 command line. The reason for this is that we actually store an AS/400 CL command string in the PPAT file and then execute it when the user presses F15.

The way you create the command string is ***almost exactly*** the same way you create the viewable message:

1. Create a data dictionary template. The only difference here is that this template is actually the shell of an interactive CL command with '&' values in place of the actual parameters that will be supplied at runtime. For example, the template for the call to the Purchase Requisition Approval program would look like this:

   ```
   CALL P43080 ('&1' '&2' '&3' '&4' '&5')
   ```

2. Run batch job J98DDMSGF. No change here.
3. Call X00PPAT1. Other that the obvious fact that this call should include information about the command string template rather that the message template, there is only one minor but critical difference to note. You must pass in the serial number that you receive back from the first call (see the output parameter description above) as an input parameter to the second call. This is so we can associate this command string with the viewable message that was just created. This link enables the message display program to know which command string to execute with a particular message.

Distribution and Manufacturing Systems

This section provides the following:

Database changes for the distribution, manufacturing, and load and delivery management systems

Distribution system changes

Manufacturing system changes

Database Changes

This chapter lists database changes for the distribution, manufacturing, and load and delivery management systems.

Distribution System

Fields Added or Changed

File	File Name	Status	Field/Notes
F4009	System Constants	Added Changed	CPCCTL ESC Control (Y/N) CPCP08 Pricing Audit (Y/N)
F40309	Delivery Date Preference File	Added	DYPRIO Priority Code
F41001	Branch Plant Constants	Changed	CIOT4Y Quality Management (Y/N)
F4211	Sales Detail File	Changed	SDPDDJ Pick Date SDPPDJ Ship Date SDSO02 Interbranch Processing (value added to data dictionary) SDSO11 Transfer/Direct Ship Order Flag SDSO12 Deferred Entries Flag
F4600	Warehouse Management Requests	Added	R1UKID Unique Key ID (Internal)
F4611	Warehouse Management Suggestions	Added	R2UKID Unique Key ID (Internal) R2CKID Confirmed Unique Key ID (Internal) R2RCDS Record Status
File	**File Name**	**Status**	**Field/Notes**

F49211	Sales Detail Tag File	Changed Added	Systems to 40/40 from 49/49 UDIAN8 Interbranch Address UDPTC Interbranch Payment Terms UDDOC Interbranch Document Number UDDCT Interbranch Document Type UDKCO Interbranch Document Company UDCRR Interbranch Exchange Rate UDCRCD Interbranch Currency Code UDTXA1 Interbranch Tax Rate/Area UDEXR1 Interbranch Tax Expl Code

Logical File Changes

File	**New Key Fields Underlined**
F41003LA	UCRUM, UCUM (record format I41003 - fields switched) UCUM, UCRUM (record format I41003I - fields switched)
F4102LM	IBPRP4, IBMCU, IBMPST, <u>IBSTKT</u>
F4102LN	IBANPL, IBMCU, IBPRP4, IBMPST, <u>IBSTKT</u>
F4102LP	IBBUYR, IBMCU, IBPRP4, IBMPST, <u>IBSTKT</u>
F4111LE	ILITM, ILMCU, ILTRDJ, <u>ILTDAY</u>
F4111LG	ILITM, ILDGL, <u>ILTDAY</u>
F4311LS	<u>PDDCTO</u>, PDAN8, PDOMCU, PDSUB, PDOBJ, PDSBL, <u>PDSBLT</u>

Manufacturing System

Fields Added

File	Data Item	Description
F3013	ITC	ITC - Issue Type Code
F3291	LITM	2nd Item Number
	AITM	3rd Item Number
	RNDC	Derived Calculation Round
	UPCC	Update Category Code
	USER	User ID
	PID	Program ID
	JOBN	Work Station ID
	UPMJ	Date Updated
	TDAY	Time of Day
F3292	LITM	2nd Item Number
	AITM	3rd Item Number
	URCD	User Reserved Code
	URDT	User Reserved Date
	URAT	User Reserved Amount
	URAB	User Reserved Number
	URRF	User Reserved Reference
F3293	LITM	2nd Item Number
	AITM	3rd Item Number
	FORQ	Fixed or Variable Quantity
	ITC	Issue Type Code
	LOVD	Leadtime Offset Days
	BSEQ	Bubble Sequence
	EPGM	External Program ID
	DERP	Smart Part Calculation
	TBLC	Rules Table Name
	URCD	User Reserved Code
	URDT	User Reserved Date
	URAT	User Reserved Amount
	URAB	User Reserved Number
	URRF	User Reserved Reference
	USER	User ID
	JOBN	Work Station ID
	UPMJ	Date Updated
	TDAY	Time of Day

F3294	CSID	Configured String ID
	USER	User ID
	PID	Program ID
	JOBN	Work Station ID
	UPMJ	Date Updated
	TDAY	Time of Day

File	Data Item	Description
F3296	LNTY	Line Type
	USER	User ID
	PID	Program ID
	JOBN	Work Station ID
	UPMJ	Date Updated
	TDAY	Time of Day

Fields Changed

File	Data Item	Description
F3294	CFGS	Configured String

Fields Deleted

File	Data Item	Description
F3294	AISL	Aisle
	BIN	Bin

New Files

File	Description

F3105	Work Order Serial Numbers
F3108	Summarized Work Order Cross Reference
F3209	Configurator Constants
F3214	Configured Model Text
F3281	Rules Table Definition
F3282	Configured Item/Rules Table Cross Reference
F32821	Rules Table Value Definition
F3283	Rules Table Detail
F32941	Configured String Master
F32942	Configured String Detail
F3405	Forecast Consumption Periods
F3462	Forecast Shipment Summary
F43211	Supplier Split Percentages

New Logicals

File	Key Fields
F3013LE	KIT, MMCU, SRV
F3013LF	ITM, MMCU, TRV
F3105LA	DOCO, DCT, MCU, LINS
F3105LB	DOCO, DCT, MCU, LOTN
F3108LA	DOCO, JDOC
F3214LA	DOCO, DCT, KCOO, LNID, LINS
F3281LA	TBLC, MCU, RTBT
F3282LA	TBLC, MCU, RTBT, KIT
F32821LA	TBLC, MCU, RTBT, KIT, BSEQ
F3283LA	TBLC, MCU, RTBT, TSV1, TSV2, TSV3, TSV4, TSV5, TSV6, TSV7, TSV8, TSV9, TSV0, BSEQ
F32941LA	KIT, CSID, SEQN
F32942LA	KIT, CSID, ITM, PEL, SGVL
F32942LB	CSID, KIT, ATLV
F32942LC	KIT, PEL, SGVL, ITM
F3411LH	ITM, DRQJ, MSGT, MSGA, HCLD
F3411LI	ITM, DRQJ, MSGT, MSGA, HCLD, TRQT, MMCU
F3462LA	ITM, MCU, PDDJ
F3462LB	ITM, PDDJ

Distribution System Changes

This chapter describes changes made to the distribution system.

Obsolete Item Processing

An item can now be marked as obsolete. This is done using the Stocking Type field (STKT) in the Item Branch Master file (F4102). The values of ‘O’ for Obsolete and ‘U’ for Obsolete - Use Up are now hard-coded in the appropriate distribution and manufacturing programs.

Receipt File Modification (F43121)

The voucher match program has been modified to update the receiver file differently for a 2-way match. During the 3-way match process, the ‘1’ match type receipt record (PRMATC = ‘1’) was created during receipts, and the ‘2’ match type receipt record (PRMATC = ‘2’) was created during voucher match. For the 2-way match process, both ‘1’ and ‘2’ match type receipt records were created during voucher match. The 2-way match process has been modified to no longer create ‘1’ match type receipt records during voucher match.

If you have ‘custom’ programs that use this type ‘1’ 2-way match record, you can identify where the match type ‘1’ records came from by using the Written By Program (PRRTBY) field. This field will contain a BLANK if the record was written by the receipts program for a 3-way match. This field will contain a ‘02’ if the record was written by the match program for a 2-way match.

If you do not have ‘custom’ programs using this record, then no conversion will be necessary. The information in the records will be updated correctly as you perform the matching process.

Exclude Cash from Open Order Amount

The order process has been enhanced so that the customer open order amount will NOT include amounts from orders that are COD or for which cash has been or will be paid.

This requires modifications to all programs that update the open order amount in the customer master record based on an amount accumulated from an order detail line. This is accomplished by using the payment instrument (RYIN). A ‘1’ in the first position of the special handling code of the UDC table 00/PY indicates this payment instrument is cash.

All programs calling X0301 to update the open order amount must be changed to exclude the cash amount. The open order amount sent to the X0301 is the difference between the total amount and the cash amount. Programs were changed to update the customer order total with a value that is kept separate from the value that is updated to order header.

For example: A 3-line order is entered:

Line #1	50.00 (credit)	
Line #2		50.00 (credit)
Line #3		50.00 (cash)

Total stored in the order header is 150.00. Customer open order total (F0301) is increased by only 100.00.

Sales Detail Tag File

The F49211 sales detail tag file is currently used by ECS. This file was changed to a system 40/40 and in A7.3 is also used by the base sales system. The base sales system will only write a record to the F49211 if the client is using the new inter-branch invoicing or invoice cycle. For ECS clients, a F49211 record will exist for every sales detail line as it does in A7.1. Code was added to check for the existence of the F49211 record in specific programs (see below).

Inter-Branch Processing

Inter-branch processing was enhanced to create an inter-branch invoice from the shipping branch to the selling branch. In A7.1, the price from the shipping branch to the selling branch was derived by the cost from the shipping branch and applying a markup. In A7.3, the client now has an option to use the base price instead of the cost plus markup.

In A7.1, the field SDSO01 was set to a ‘1’ to indicate the order was an inter-branch order. Programs checked this field equal to a ‘1’ or not equal to a ‘1’.

```
  SDSO01  IFEQ  ‘1’       or     SDSO001 IFNE ‘1’
```

The values of SDSO01 were expanded to include a 2, 3, and 4. If the field is non-blank, the order is classified as an inter-branch order. This field is now used to indicate what type of processing this order will go through. The meaning of the values are as follows:

blank	Order is NOT an inter-branch order
1	No inter-branch invoicing/cost plus markup
2	Inter-branch invoicing/cost plus markup (F49211 record written)
3	No inter-branch invoicing/price
4	Inter-branch invoicing/price (F49211 record written)

Programs updating the cost by calling XF4105 will need to be changed to check if the SDSO01 flag is set to ‘1’ or ‘2’.

```
SDSO01    IFEQ ’1’
SDSO01    OREQ ’2’
          CALL S005T   (subroutine which calls the XF4105)
          ---- ------
          ENDIF
```

In specific cases, programs were altered to retrieve/add the Sales Detail tag file (F49211). These programs were P42565, P42800, P42997, P4205, P42950, and order entry programs. If you need to use a field in the file for purposes of inter-branch sales, the F49211 file must be retrieved by checking that the SDSO01 flag is set to ‘2’ or ‘4’.

Invoice Cycle

The invoice cycle enhancement allows the invoice frequency to be specified at an item, customer, or customer/item level. When the invoice cycle program (P49700) runs, a sales detail tag file (F49211) is written if it does not exist. The SDSO12 flag is used to indicate if deferred entries were written and, subsequently, a F49211 record. If you need to use a field in the file for purposes of invoice cycle, the F49211 file must be retrieved by checking that the SDSO12 flag is set to a ‘1’.

The following example is to retrieve the F49211 record:

```
SDSO01    IFEQ ’2’
SDSO01    OREQ ’4’
SDSO12    OREQ ’1’

$49211    IFEQ ’ ’
          OPEN F49211                  81
*IN81     IFEQ *OFF
          MOVE ’1’      $49211  1
          END
          END
```

Pricing Security - Audit Log

The price files F4070, F4071, F4072, F4075, F4076, and F4106 now have a corresponding audit file (F4070A, F4072A, and so on) that is written to, whenever a record in these files is changed. A record is written to the audit file before the change, and then a record is written to the audit file with new data. Programs updating any of these files are changed to write to the audit file if the Pricing Audit (Y/N) flag is set to ‘Y’. This flag is located on the pricing constants window.

Sales Order/Purchase Order Integrity

Currently there is no way to ensure the integrity of related sales and purchase orders created during the Transfer Order procedure. This procedure creates related sales and purchase orders. However, changes to these orders in their respective systems do not affect the related order. Changes to the related order have to be done manually, creating duplication of labor and adding to the possibility of keying errors.

The programs to be modified for sales order/purchase order integrity will be:

Transfer Order Entry (P4242)

Enter Sales Order (P4211)

Enter Sales Order - Line Mode (P4201A)

Inventory Availability/Commitment (P42997)

Ship Confirm (P4205)

For Transfer Order Entry (P4242), a flag will be set in the purchase order and the sales order detail records to identify inventory detail lines that are generated by this procedure. These flags will be used as a trigger in the Sales Order System to update the related purchase order.

To update a sales order generated by Transfer Order Entry (P4211), a server will be called to update the related purchase order.

To update a sales order detail line generated by Transfer Order Entry (P4201A), a server will be called to update the related purchase order detail line.

For Inventory Availability/Commitment (P42997), for any sales order generated by Transfer Order Entry, a server will be called to update the related purchase order for any changes, including the splitting of detail lines.

To confirm a sales order generated by Transfer Order Entry (P4205), a server will be called to update the related purchase order with the information entered in the confirmation process.

Conceptual Approach

For sales and purchase orders generated by Transfer Order Entry, a flag will be set in both detail records to mark these records as being generated from these procedures. Also, any procedure in the Sales Order System that updates these sales orders will call a server to update the related purchase order. The fields to be updated include branch plant, quantities, cost amounts, lot numbers, serial numbers, unit of measures, and prices.

The following logic flow describes the Enter Sales Order procedure.

1. The client goes into the Enter Sales Order procedure and updates the sales order.
2. After the sales order has been updated, a server is called and the related Purchase Order is updated.

The following logic flow describes the Enter Sales Order - Line Mode procedure.

3. The client goes into the Enter Sales Order - Line Mode procedure and updates a detail line.
4. After the sales order detail line has been updated, a server is called and the related purchase order detail line is updated.

The following logic flow describes the Ship Confirm procedure.

5. The client goes into the Ship Confirm procedure and follows the normal confirmation process.
6. After the sales order has been updated, a server is called and the related purchase order is updated, including the mirroring of any split detail lines on the sales order.

Database Specifications

The use of field PDPS01 in the Purchase Order detail file (F4311) and field SDSO11 in the Sales Order detail file (F4211) will be updated for line items generated by the Sales Transfer Order procedure. These flags will enable the update programs in the Sales Order System to identify which sales orders have related purchase orders to be updated.

Program Specifications

Program	**Menu Line Title**	**Status**
P4242	Enter Orders (Transfer)	Changed
P4211	Enter Sales Order	Changed
P4201A	Enter Sales Order - Line Mode	Changed
P42997	Inventory Availability/Commitment	Changed
P4205	Ship Confirm	Changed
X4250	Update Related Purchase Order	New

Special Logic

This is a new server to be called when a sales order generated by the Transfer Order procedure is updated. This server will update the related purchase order with the changes to the inventory detail lines of the sales order. This is determined by flag SDSO11 in the sales order detail record being set to ‘1’.

Summary Matching/Re-Costing

There are two issues that some companies encounter when processing invoices.

The first issue is the receiving of an adjustment invoice for product that was already invoiced. This may occur due to a price change that affected the product shipped to the company. This also may be a correction amount for the initial invoice that was billed in error. In either case, the company needs to adjust the previous voucher to reflect the new cost.

The second issue is that some companies process large volumes of transactions in a given period. They may, however, receive summarized invoices for numerous transactions. Matching these individual receipts to the summarized line on the invoice is a slow and error-prone task.

Objectives

The objective of this program is to provide the following functions:

To provide the ability to create an adjustment invoice for a receipt that has already been vouchered. This need normally arises from the company receiving an adjustment invoice from a vendor after the original invoice was received.

To provide a program that allows the user to summarize receipt lines into a single line when there is a high-transaction rate over periods of time. These are companies who process large volumes of transactions and receive summarized invoices that are extremely difficult to match against detail receipt records.

Scope

This issue will require the following new programs as well as modifications to existing programs:

Program	Title	Status

P4315	Summary Voucher Match	New Program
J4315	Summary Voucher Match CL	New Program
DTAQBV	Summary Batch Voucher Data Queue	New Object
F43800	Summary Match Work File	New File
J43800Q	Summary Voucher Match CL	New Program

See the detail designs for each of the following programs for specific impact.

Program	**Title**	**Status**
P43800	EDI File Feeder program	Modification
P470412	EDI Voucher Match	Modification
P4314	Detail Voucher Match	Modification

Conceptual Approach

Display

The information may be *displayed* in a summary or detail format. In detail display mode, each receipt or voucher is displayed as an individual subfile line. In summary display mode, the receipts are summarized by item, company, currency code, unit of measure, subledger, and subledger type into one subfile line.

Process (Match) Type

The receipts displayed can be receipts that have not been vouchered or receipts that have already been vouchered. The receipts that have not been vouchered (type 1) will be matched with the invoice and closed. The receipts that have been vouchered will create an adjustment voucher that is associated with the original receipt.

Matching (Vouchering)

Once the information displayed is verified, then the subfile lines may be *matched* in detail or summary mode. In detail mode, the program would simply call the P4314 (Detail Voucher Match) program for each receipt or voucher that exists for the subfile line to be processed. If the display mode is summary, then the program will call the P4314 program for every receipt record that was summarized into the subfile line.

In summary mode, the program would first flag **each** receipt summarized in the subfile record with a processed flag (ILOG) of ‘P’ and write the new cost information to the work file (F43800).

An entry that contains the version of program P43800 to call will be written to a data queue (D43800). The data queue will activate the sleeper program J43800Q, which will call P43800.

The P43800 program will then read the receipts file for all records flagged with ‘P’ in the processed field. It will then chain to the F43800 file to retrieve the new cost information for that receipt. Once the new information is received, records will be written to the EDI files F47041 and F47042 to represent the voucher.

Once all records are processed, the P43800 program will call the P470411 program to match the receipts.

Database Specifications

Existing Files

File	File Name
F43121	Purchase Order Receiver File
F43121LD	Purchase Order Receiver File - Logical File - Match Type, Document Number, Document Type, Document Company, Business Unit
F43121LF	Purchase Order Receiver File - Logical File - Match Type, Supplier Number, Item Number, Business Unit, Receipt Date

New Files

File	File Name
F43121LJ	Purchase Order Receiver File - Logical File - Match Type, Supplier Number, Contract Number, Contract Number, Item Number
F43121LK	Purchase Order Receiver File - Logical File - Match Type, Supplier Number, Supplier Remark, Item Number
F43800	Voucher Match Workfile

Serial Number Processing

With the continually increasing levels of regulation, consumer liability, and quality assurance demands, a growing number of companies are requiring the ability to identify and trace serialized components and assemblies throughout the manufacturing and distribution process.

The approach of the enhancement is to build on the existing lot number functionality. Serial numbers will be processed as lots with a quantity of one, which is considered the lowest trackable unit. Tracing and tracking the serial number throughout the system and beyond will be done through the current lot trace/track program. New fields will be added to enable serialized items to be under lot control also.

Term Definitions

Term	Definition
LSN	The item's lot/serial number. This is the lowest trackable unit. Throughout the *Programmer's Guide*, we will use this term to refer to the lot or serial number.
Lot Number (LOTN)	A number assigned to a group of items for identification.
Serialized Tracked Item	This implies that the item has an LSN and has been set up so that the quantity for the item may not exceed one.
Lot-Tracked Item	This implies that the item has an LSN and has been set up so that the quantity for the item may exceed one.
Non-Tracked Item	An item that does not have an LSN.
Memo Lot 1	A higher classification or grouping of serialized items. For example, a group of items will have unique serial numbers but the same lot number.
Memo Lot 2	A higher classification or grouping of memo lot 1.
Memo Lot 3	A higher classification or grouping of memo lot 1 and memo lot 2.

Term	Definition
Supplier Lot	A lot number the supplier assigned to this item.
Lot Process Type (SRCE)	A code to indicate whether lot assignment is to be used and the method to assign to the lot number. The current allowed values are 0-3. Values of 4-7 will be added for this project. 0 LSN optional; quantity can be greater than one 1 LSN generated in YYMMDD format; quantity can be greater than one 2 LSN generated by next numbers; quantity can be greater than one 3 LSN required, must be manually assigned; quantity can be greater than one 4 LSN optional; required during shipment confirmation, but optional throughout the rest of the system; quantity must not exceed one (only when an LSN is entered) 5 LSN generated in YYMMDD format; quantity cannot be greater than one 6 LSN generated by next numbers; quantity cannot be greater than one 7 LSN required, must be manually assigned; quantity cannot be greater than one Option 4 will enforce quantity restrictions if the LSN is entered. If the LSN is not entered, the information will be processed as if the SRCE value is '0'.
Serial Number Required (SRNR)	This field, in the current J.D. Edwards system, controls whether an item requires a serial number to be entered during shipment confirmation. These values will remain the same to provide consistency with the current J.D. Edwards system and will be referred to as basic serial number processing. This design document will only address the new advanced serial number processing. Three new values will be added to this data item to control memo and supplier lot functionality for the advanced serial number processing. 3 Supplier lot number required (Purchasing only). 4 Supplier lot number required (Purchasing only). Lot 1 required (Purchasing, Sales Order Processing, Manufacturing, and Inventory). 5 Supplier lot number required (Purchasing only). Memo Lot 1 required (Purchasing, Sales Order Processing, Inventory, and Shop Floor Control). Memo Lot 2 required (Purchasing, Sales Order Processing, Inventory, and Shop Floor Control). N No serial number processing. Y Basic serial number processing.

Information Structure

Purchasing

Purchase Order Receiver File (F43121)

There are no changes to this file. It will contain a record for each item with a serial number attached to it. The quantity will be restricted to no more than one when the rules dictate it.

Purchase Order Detail (F4311)

There are no changes to this file.

Inventory Management

File	File Name	Table Description
F4101	Item Master	There are no changes to this file. The LSN designation can be made here to default into the Item Branch records. This applies to the Lot Process Type and Serial Number Required fields also.
F4102	Item Branch	There are no changes to this file, but the Lot Process Type and Serial Number Required fields will be looked at here by all appropriate programs for LSN processing.
F41021	Item Location	There are no changes to this file. It will contain a record for each item with a serial number attached to it. The quantity will be restricted to no more than one when the rules are set accordingly.
F4108	Lot Master	Three new fields were added to maintain related lot information for serial number items. The three new fields are Memo Lot 1, 2 and 3 and were added in the A7.1 release. All three of these fields will be maintainable using the Lot Master Revisions program (P4108). Memo Lot 1 or Memo Lot 1 and 2 may be required throughout the transaction programs, depending on the Serial Number Required field (SRNR) for the item.

F4111	Item Ledger	There are no changes to this file. Each time an inventory transaction takes place for an item assigned a serial number a record will be written to this file.

Shop Floor Control

Work Order LSN (F3105)

A new file will be added that contains fields to identify work order assemblies with LSN’s.

Sales Order Processing

Sales Order Detail (F4211)

There are no changes to this file. It will contain a record for each item with a serial number attached to it.

Issues and Assumptions

The Lot Number (LOTN) field will be used for both serial and lot numbers depending on tracking required for individual items.

We will display only 12 characters of the LOTN field for allowing entry of either the serial number or lot number.

We will only display and use the Memo Lot 1 and Memo Lot 2 fields. The exception to this will be on the Lot Master maintenance video where all three will be displayed and maintained. Memo Lot 3 will be retained for future use.

All current lot processing functions (such as lot expiration date, lot status code, duplicate lots allowed, and so on) will continue to function over the LSN. This will require an expiration date for all LSN-controlled items.

Sales Order Processing/Purchasing

The original serial number processing for purchase order and sales order processing will continue to exist and will be referred to as basic serial number processing. You cannot do both the basic and advanced serial number processing simultaneously on an item.

During sales order entry, if an item is serialized, the transaction unit of measure *must* be in primary.

A Sales Order Detail file record will exist for every serial number sold.

The on-hand flag will determine when time a serial number must be entered. This means that a serial number must be attached to a transaction before the process can be moved to on-hand. However, in receipt routing, the serial number can be assigned during any step in the process.

In the purchasing system, an item can be entered into the system as you can currently do today, but when an item is received, whether through receipt routing or not, the receipt *must* be entered by using the Primary Unit of Measure field when the item allows serial numbers.

In purchasing, when processing a serial number required item with multiple quantities, the receipt program and movement/disposition program will require the user to split the lines into single-quantity lines when either assigning a serial number or moving the quantity to on-hand.

A purchase order receiver record will exist for every serial number moved to an on-hand status by using the receipts program or the receipts routing process.

Inventory

An item should not be serial-number controlled in one branch/plant or warehouse and lot controlled in another branch/plant or warehouse. Within the inventory management transaction programs, the client will be forced to enter a transaction line for each serialized number they desire activity on.

Shop Floor Control

LSNs can be assigned to specific assemblies at any time before work order completion by using the Assign Work Order LSN program. Assembly LSNs can also be assigned at the time of work order completions by using the Associate Issued Item LSNs program.

Work orders for serialized assemblies must be entered in their respective primary unit of measure to allow for assignment of serial numbers.

Serialized components of an assembly can be associated to a specific assembly LSN either at inventory issues or work order completion time. Serialized components must be issued in their respective primary unit of measure to allow for association.

Although changes were made to many programs, only programs commonly called are listed below.

XT4102Z1 - Item Balance Functional Server (Changed)

Program Purpose

This server will be modified to process the LSN based on the new SRCE values and perform the additional edits. The item's SRCE value will be checked when processing an item. If the item's SRCE value is a 4, 5, 6, or 7, this program will ensure that the resulting quantity of the transaction being performed does not exceed one. This quantity check should occur across all locations and branch plants (see X41QTY).

Nature of Change

All current edits being performed over the LOTN field based on the current SRCE values should include the new SRCE values.

All editing for SRCE value 0 should include SRCE 4.
All editing for SRCE value 1 should include SRCE 5.
All editing for SRCE value 2 should include SRCE 6.
All editing for SRCE value 3 should include SRCE 7.

When processing an item with a SRCE value of 5-7, or when the item's SRCE value is 4 and a value has been entered in the LOTN field, the X41QTY server should be called. The item number, MCU, and transaction quantity should be passed to this server. This sever will then determine if the quantity for the transaction will result in a quantity for the item's LSN to be greater than one. Both this and the X41QTY server must ensure that inventory transfers are processed properly. The appropriate error message should be issued.

The following list represents all programs calling this server. These programs will be affected by the server changes but will not need to be changed:

P4112	Simple Issues
P4113	Transfers
P4114	Adjustments
P4116	Reclassification

X41QTY - LSN Quantity Check (New)

Purpose of Program

This server will be used to check item locations that are under LSN control to ensure that transactions manipulating inventory quantities do not result in a quantity for the item that exceeds one (1). This server will check the existing quantity for an item's LSN across all branch plant locations marked as quantity restricted. This program will ensure that the quantity in the system, plus the quantity of the transaction for the LSN, cannot exceed one (1).

A flag will be returned to the calling program to indicate if the LSN being processed already exists in the system for this item. This flag can be used in the manufacturing, purchasing and the warehouse management systems to ensure that a depleted LSN under quantity restriction does not get replenished.

Program Type

Server routine.

Special Logic

When calling this program, two errors parameters are returned. The first should be a hard error that indicates that a quantity restrictions attempt violation. The second error should be considered a ‘soft’ error. Some programs have processing options (indicating restrictions on item replenishment) that will indicate if this additional error parameter should be checked. If a program does not contain this processing option, this error parameter should be ignored.

Significant File Usage

F41021	Item Location file

X4108- LSN Master Update (Changed)

Program Purpose

This server will be modified to update the new Memo Lot fields in the Lot Master file from the calling programs. All programs that are used by this server will need the Memo Lot fields added to the data structures. If a program allows entry of memo lot information, those values will need to be moved into the program data structure so that this routine can create the LSN Master records with the correct information. This functionality will only be available for the creation of the LSN master records.

Nature of Change

The following programs call this server and will need to be changed to pass in the new Memo Lot fields in the data structure:

Program	Name

P31113	Work Order Inventory Issues
P31114	Work Order Inventory Completion
P31115	Co/By Product Completion Window
P31123	Super Backflush
P3114	Rate Schedule Workbench
P31143	Rate Base Inventory Issues
P31420	Work Order Automatic Batch Issue
P41082	Hold Expired Lots
P41280	Lot Availability
P41413	Update Cycle Count
P4141	Cycle Count Entry
P41021W	Primary Location Window
P41024	Item Location Information
P4108	Lot Master Revisions
P4108S	Location Lot Status Revisions Window
XT4102Z1	Item Balance Server
P41602	Tag Inventory Count Entry
P41610	Tag Inventory Update
P4205	Shipment Confirmation
P42800	Sales Update
P4312	Receipts by PO/Item/Account
P43250	Routing Movement
P48013	Manufacturing Work Order Entry
P49800	Sale Update - ECS Version

X41LOT - LSN Assignment (Changed)

Program Purpose

This program will generate LSNs for new balance records based on the SRCE value for the item. This routine will be enhanced to generate the LSN based on the new SRCE values being added to the system.

Nature of Change

The program currently processes with A5 logic for the SRCE field, and uses the values ‘6’ and ‘7’ for SRCE. This code will be removed from the program.

The program currently generates an LSN based on the SRCE value of ‘1’ or ‘2’. This routine will be modified to include the new SRCE values (SRCE = ‘5’ and ‘6’). All processing for SRCE ‘1’ and ‘2’ will be performed for SRCE ‘5’ and ‘6’ respectively.

P42053 - Multiple Location Window (Changed)

Program Purpose

This program currently contains two formats. One format is used during the sales cycle and the second is used during the purchasing cycle. The new Memo Lot 1 and 2 fields and the Vendor Lot field will also be added to both formats. The LSN expiration date and status code will also be added to the sales format.

When called from the purchasing programs, the calling programs should ensure proper editing is performed to ensure that the item's LSNs, Vendor Lot, and Memo Lot 1 information are entered (if required). This editing information will be retrieved from the item's SRCE and SRNR values.

When called from ship confirmation, this program will allow the user to record the LSN, vendor lot, and Memo lot 1 and 2 values for the first time for items containing a SRCE value of four (4 - LSN optional with a quantity restriction) and/or an SRNR value of 1 through 3.

Nature of Change

Add the Memo Lot 1 and 2 and Vendor Lot fields to both formats. Add the LSN expiration date and status code to the second format. For the purchasing format, the lot fields will be in the header and will default to the subfile lines in which no lot field information has been entered.

The following logic should be performed for inventory receipts. When entering an item's LSN, the new SRCE values should be included for the edit checks.

All editing for SRCE value 0 should include SRCE 4.
All editing for SRCE value 1 should include SRCE 5.
All editing for SRCE value 2 should include SRCE 6.
All editing for SRCE value 3 should include SRCE 7.

When processing an item with a SRCE value of 5-7, or when the item's SRCE value is 4 and a value has been entered in the LOTN field, the X41QTY server should be called. The item number, MCU, and transaction quantity should be passed to this server. This server will then determine if the quantity for this transaction will result in a quantity for the item's LSN to be greater than one. The appropriate error message should be issued from this program.

The item's SRNR value should be checked and based on its value required by the Vendor Lot and Memo Lot 1 and 2 values.

Currently a value is passed to this program from the calling program to indicate what the calling program is (purchasing, sales, manufacturing, and so on). The multi-locations program will process data with a slight variation based on this value. A new value has been added ('D') to the purchasing system. This value should have all the same functionality as the purchasing value ('P'). In addition, the total quantity of the item to be selected in the multi-location window must equal the actual quantity selected. Currently a 'P' allows under-selection. This new functionality will only be called during the routing cycle when the user is attempting to assign an LSN.

The following logic should be performed for ship confirmation:

When processing items with a SRCE = 4, in addition to the current functionality, allow entry of non-existent LSNs for existing branch plant locations of the item. The item's SRCE value must be a '4' to allow this processing. This will allow entry of LSNs during the shipment cycle. This functionality is similar to what occurs at inventory receipts time. This program should require entry for the Vendor and Memo Lot fields when applicable.

The following programs need to be modified (extend the user index size) so that the multi-location window can continue to work with them:

Program	**Name**
P3111	**Work Order Parts List Revisions**
P31113	**Work Order Inventory Issues**
P31114	**Work Order Inventory Completion**
P3114	**Rate Schedule Work Bench**
P31143	**Rate Base Inventory Issues**
P4205	**Shipment Confirmation**
P4312	**Receipts by PO/Item/Account**
P43250	**Routing Movement**

By way of a function key, a selection criteria window can be called. The selection window will display memo lot 1 and 2, vendor lot, LSN range, and lot status (inventory). This enables the user to narrow the scope of the LSNs displayed in the multi-location window.

Function Keys/Option Selections

Function Key 6 = Selection Criteria Window

P40SEL - Selection Criteria (New)

This window will display: memo lot 1 and 2, vendor lot, LSN range, and lot status (inventory only). A parameter is passed to this program to indicate if the calling program is a purchasing, sales, inventory, manufacturing, or warehouse program. The program will pass back any values entered in the window to be used by the calling program to display only the LSNs that pass the criteria.

The following programs will access this window:

Multi-Location (P42053)
Item Summary (P41202)

P40ITM2 - Item Selection Window with Quantity (Changed)

Purpose of Program

This program is used to select items from multiple locations and return the quantities selected to the calling program. Some additional fields will be added to allow the user to vary the way items are viewed/selected. This window program is called from a number of applications. These applications must have their processing options set to enable this program.

Nature of Change

The video will have the following new fields added to it. These fields will allow the user to view/select items by using different search criteria.

Fields	**Description**
Item Number	Display all locations for an item. The search window will also call P40SEL through a function key for the additional search criteria.
Memo Lot 1	Display only those LSNs for this Memo Lot 1 value (LSN mode only).
Memo Lot 2	Display only those LSNs for this Memo Lot 2 value (LSN mode only).
LSN Selection	Range of LSNs to be selected (LSN mode only).

Stock Valuation

Stock Valuation provides the tools for companies to effectively determine their stock values for reporting and evaluating their profit margins. Stock valuation is calculated monthly, quarterly, or yearly.

Objectives

The objectives of this module are to provide the following functions:

Measure and manage stock levels and related cash flow.

Comply with accounting standards that require a company to provide a true and fair value of the company's financial performance and capitol employed.

Scope

These objectives require the following new programs:

Program	Title
P39001	Item/Pool Inquiry
J39001	Item/Pool Inquiry CL
V39001	Item/Pool Inquiry Video
P3901	Item/Pool Valuation Maintenance
J3901	Item/Pool Valuation Maintenance CL
V3901	Item/Pool Valuation Maintenance Video
P3902	Valuation Method Master
J3902	Valuation Method Master CL
V3902	Valuation Method Master Video
P3902W	Valuation Method Window
V3902W	Valuation Method Window Video
P3905W	Period Additional Quantities
J3905W	Period Additional Quantities CL
V3905W	Period Additional Quantities Video
P39050	Valuation Summary Review
J39050	Valuation Summary Review CL
V39050	Valuation Summary Review Video

P39051	Valuation Period Review
J39051	Valuation Period Review CL
V39051	Valuation Period Review Video
P39052	Valuation Layers Review
J39052	Valuation Layers Review CL
V39052	Valuation Layers Review Video

Program	**Title**
P39053	Period Summary Review
J39053	Period Summary Review CL
V39053	Period Summary Review Video
P3906	Document Summary Review
J3906	Document Summary Review CL
V3906	Document Summary Review Video
P39120	Valuation Period Extraction - Workfile Build
P391201	Valuation Period Extraction
J39120	Valuation Period Extraction CL
R39120	Valuation Period Extraction Print File
P39130	Valuation - G/L Update
J39130	Valuation - G/L Update CL
R39130	Valuation - G/L Update Print File
P39200	Valuation Method Comparison
J39200	Valuation Method Comparison CL
V39200	Valuation Method Comparison Video
P39400	Stock Valuation Detail Report
J39400	Stock Valuation Detail Report CL
R39400	Stock Valuation Detail Report Print File
P39500	Valuation Summary Report
J39500	Valuation Summary Report CL
R39500	Valuation Summary Report Print File
P39510	Valuation G/L Update Summary
J39510	Valuation G/L Update Summary CL
R39510	Valuation G/L Update Summary Print File

P39540	Unit Cost Period Report
J39540	Unit Cost Period Report CL
R39540	Unit Cost Period Report Print File
P39900	Valuation File Purge
J39900	Valuation File Purge CL
R39900	Valuation File Purge Print File

Database Specifications

New files

File	File Name	Prefix
F3901	Item/Pool Valuation Master	PL
F39011	Company G/L Update Method	SC
F3902	Valuation Method Master	SV
F39051	Valuation Period	SO
F390519	Valuation Period - Purged Records	SO
F29052	Valuation Layer	ST
F39053	Period Additional Quantities	SU
F390539	Period Additional Quantities - Purged Records	SU
F3906	Valuation Document Summary	DS
F39069	Valuation Document Summary - Purge Records	DS
F3911	Item Ledger Tag	SL

New Logical Files

File	Key Fields
F3901LA	CO, IPTK, ITM, SVVM
F3901LB	SVVM, CO, ITPL, ITM
F39051LA	SVVM, ITPL, ITM, CO, MCU, FY, PNC
F39051LB	CO, MCU, SVVM, FY, PNC, ITPL, ITM
F39051LC	FY, PNC, SVVM, CO, ITM, ITPL, MCU
F39051LD	FY, PNC, CO
F39051LE	SVVM, ITPL, ITM, MCU, FY, PNC
F39051LF	CO, ITPL, ITM

F39052LA	SVVM, FY, PNC, ITPL, ITM, CO, MCU, RCPD, RCSQ
F39052LB	FY, PNC, CO
F39052LC	SVVM, ITPL, ITM, CO, MCU, RCPD, RCSQ
F39052LD	SVVM, ITPL, ITM, CO, RCPD, RCSQ
F39052LE	FY, PNC, CO
F39053LA	FY, PNC, CO
F3906LA	FY, PNC, CO
F3906LB	FY, PNC, DCT, ITPL, ITM, CO, MCU
F3911JA	F4111/F3911

Container Management

Container Management is used to manage the inventory of containers as they are exchanged between the customer and the supplying company.

The sale of products in containers have a unique inventory process that requires an enhanced inventory management system compared to the standard Distribution industry systems. The containers are not sold to the customers, only the product in the containers are sold. Hence, the container is only loaned to the customer for the storage of the product sold. A customer may have one or more containers in their possession based on their need and application and will periodically return the empty containers to the company in exchange for full containers.

Objectives

The objectives of this module are to provide the following functions:

Allow the company to account for the containers that they own. This is imperative, since the containers are of high value to the company, ranging from three to four times the cost of the product sold in them.

Track and manage the deposits and rental fees per container as charged for each customer.

Scope

These objectives require the following new programs:

Program	Title
P4118	Container Deposit Inquiry
J4118	Container Deposit Inquiry CL
V4118	Container Deposit Inquiry Video
P41180	Container Billing
J41180	Container Billing CL
R41180	Container Billing Print File
P41181	Container Transaction Inquiry
J41181	Container Transaction Inquiry CL
V41181	Container Transaction Inquiry Video
P41182	Customer/Distributor Balance
J41182	Customer/Distributor Balance CL
R41182	Customer/Distributor Balance Print File
P41185	Container Reconciliation
J41185	Container Reconciliation CL
R41185	Container Reconciliation Print File
P41189	Container Extraction Server
J41189	Container Extraction Server CL
R41189	Container Extraction Server Print File

Database Specifications

New files

File	File Name	Prefix
F4118	Container Deposit	CN
F41181	Container Transaction	CM
F411819	Container Transaction History	CM
F41185	Container Reconciliation	CT
F41189	Container Deposit History	CN

New Logical Files

File	Key Fields

F4118LA	AN8, RORN, RCTO, RKCO, RLLN, ITM
F41181LA	AN8, ITM, DOCO, DCTO, KCOO, LNID
F41181LB	AN8, ITM, LNTY, SIDT
F41181LC	ITM, MCU, RCFL
F41181LD	AN8, RORN, RCTO, RKCO, RLLN, ITM

Load/Delivery and Agreement Management Systems

Database Changes

New files

File	File Name	File Contents	Prefix
F38001	Agreement Management Constants	Agreement Management Constants	DC
F38010	Agreement Master	Agreement Master	DN
F38011	Agreement Quantities	Agreement Quantities	DF
F38012	Product Source/Destination Master	Product Source/Destination Master	DP
F38013	Agreement Quantities Schedule	Agreement Quantities Schedule	DQ
F38014	Agreement Formulas and Factors	Agreement Formulas and Factors	DR
F38111	Agreement Transaction Ledger	Agreement Transaction Ledger	DZ
F38112	Agreement Committed Quantities	Agreement Committed Quantities	DX
F49110	Actual Trip Detail	Actual Trip Detail	TA
T49551	Manual Document Temporary Workfile	Manual Document Temporary Workfile	TK

New Logical Files

File	Key Fields
F38010LA	DL01, DMCT, DMCS
F38010LB	DMTC, DMCT, DMCS
F38010LC	DMSC, DMCT, DMCS
F38010LD	AN8, DMCT, DMCS
F38010LE	AN8, DMCT, DMCS
F38010LF	ZD02, DMCT, DMCS
F38010LG	ZD03, DMCT, DMCS
F38010LH	ZD04, DMCT, DMCS
F38010LI	PANM, DMCT, DMCS
F38010LJ	PAGM, PAGS, DMCT, DMCS
F38011LA	DMCT, DMCS, DTO, ITM, DES, DESY, EFTJ
F38011LC	ITM, DMCT, DMCS
F38012LC	ITM, DTO, PSR, PSRY, DES, DESY, EFTJ
F38012LD	PSR, PSRY, DMCT, DMCS, SEQ
F38012LE	DES, DESY, DMCT, SEQ, PSR, PSRY
F38111LA	DMCT, DMCS, SEQ, PSR, PSRY, TRDJ
F49110LA	VMCU, TRP, DOCO, DCTO, KCOO, LNID
F49511LH	VMCU, TRP, LCMP, ITM, TRRD
F49511LD	VMCU, TRP, ITM, DOCO, DCTO, ITM, TRRD
F49511LF	VMCU, TRP, TRRD, DOCO, DCTO, ITM
F49511LG	VMCU, TRP, ITM, TRRD, DOCO, DCTO, KCO, CMPN

Fields Added

File Names	Field Description	Field Name
T49510	Ambient Volume	AMBR
	Standard Volume	STOK
	Volume UOM	BUM3
	Weight Result	WGTR
	Weight UOM	BUM5
	Number of Order Lines	NLIN

T49710	Ambient Volume	AMBR
	Standard Volume	STOK
	Volume UOM	BUM3
	Weight Result	WGTR
	Weight UOM	BUM5
	Number of Order Lines	NLIN
	Unit of Measure	UOM
	Volume Correction Factor	VCF

File Names	**Field Description**	**Field Name**
T49715	Load Item Type	LDIT
	Order Number	DOCO
	Order Type	DCTO
	Line Number	LNID
	Order Company	KCOO
	Business Unit	MCU
	Disposition Code	DSCD
	Promised Delivery	PDDJ
	Unit Cost	UNCS
F41011	Minimum Volume Correction Factor	MNVC
	Maximum Volume Correction Factor	MXVC
F49021	Volume Correction Factor	VCF
F4912	Volume Correction Factor	VCF

F49211, F49211Z	Foreign or Domestic	FRDM
	Port	FUPT
	Aircraft/Marine Registration	RINO
	Number (for example, tail	
	number)	GLOC
	Gate Number/Bay Number	
	/Dock Number /Berth	AUTA
	Authorization Name	ALPH
	Ships Name	MET1
	Meter Ticket Number 1	OPN1
	Beginning Meter Reading 1	PNR1
	Closing Meter Reading 1	MET2
	Meter Ticket Number 2	OPN2
	Beginning Meter Reading 2	PNR2
	Closing Meter Reading 2	MET3
	Meter Ticket Number 3	OPN3
	Beginning Meter Reading 3	PNR3
	Closing Meter Reading 3	ARDT
	Expected Arrival Date	ARTM
	Expected Arrival Time	DPDT
	Expected Departure Date	DETM
	Expected Departure Time	DSTJ
	Date Fueling Started	STM
	Time Fueling Started	END
	Date Fueling Ended	ETM
	Time Fueling Ended	

Data Dictionary

Fields Changed

Data Item Name	Description	Change
DSNN	Destination	Added UDC. Field has been in the database for some time. The base package software will be using these fields for the first time in A7.3. While the fields are longer than 3 positions, only 3 positions will be displayed and printed.
SORG	Origin	Added UDC. Field has been in the database for some time. The base package software will be using these fields for the first time in A7.3. While the fields are longer than 3 positions, only 3 positions will be displayed and printed.

DATY	Allowed Values	Allowed Values include W, V, and S.

File Servers

X3811 - Agreement Search/Edit Server (New)

Purpose of Program

This program will perform two major functions within the Agreement Management system. In search mode, it will perform a contract search based on the parameters passed from the calling program. In edit mode, it will edit the agreement and contract supplement.

Parameters

The following parameters will be used to receive data from and send data to the calling program. Note that the bulk of the parameters will be defined within a 200-byte data structure to better allow for additional parameters being added at a later date.

Parameter	**Field Size (Type)**	**Description**
PSPID	10 (A)	Program ID (X3811)
PSVERS	10 (A)	Program version
PSMODE	1 (A)	Program mode; possible values are 1 (search mode) and 2 (edit mode)
PSPARM	200 (A)	Data structure containing the following fields
PSDTO	1 (A)	Due To to be used for the contract search/edit
PSITM	8,0 (S)	Item (short format)
PSSRC1	12 (A)	Source 1
PSSRT1	2 (A)	Source type 1
PSSRC2	12 (A)	Source 2
PSSRT2	2 (A)	Source type 2
PSDES1	12 (A)	Destination 1
PSDST1	2 (A)	Destination type 1
PSDES2	12 (A)	Destination 2
PSDST2	2 (A)	Destination type 2
PSDTE	6,0 (S)	Date (Julian) to be used for contract search/edit

PSQTY1	15,0 (S)	Quantity to be used for contract search/edit
PSUOM	2 (A)	Unit of measure of QTY1
PSSRCH	1 (A)	Contract search processing option
PSDMCT	12 (A)	Contact number
PSDMCS	3,0 (S)	Contract supplement
PSSEQ	6,0 (S)	Sequence
PSUPRC	15,0 (S)	Override price
PSASN	8 (A)	Price schedule
PSUPGL	1 (A)	Update G/L
PSPRAS	1 (A)	Update at ambient/standard/weight
PSERR	4 (A)	Error message ID
Parameter	**Field Size (Type)**	**Description**
PSDOCO	8,0 (S)	Order number
PSDCT	2 (A)	Order type
PSKCOO	5 (A)	Order company
PSLNID	3,0 (S)	Line number

XT38111 - Agreement Transaction Server (New)

Purpose of Program

The purpose of this program is to perform all updates in the Agreement Management system related to agreement balances and contract transactions. This program will be called by the Sales Order Entry, Purchase Order Entry/Receipts, Load Confirmation, and General Stock Movement programs.

Parameters

The following parameters will be used to receive data from and send data to the calling program. Note that the bulk of the parameters will be defined within a 200-byte data structure to better allow for additional parameters being added at a later date.

Parameter	Field Size (Type)	Description
PSPID	10 (A)	Program ID (X38111)

PSVERS	10 (A)	Program version
PSMODE	2 (A)	Program mode; a code that represents the function to be performed
PSERR1	4 (A)	Error message ID
PSERR2	4 (A)	Error message ID
PSPARM	200 (A)	Data structure that contains the following fields

The next five fields represent the unique key to a F38012 record for purposes of updating commitment quantities.

Parameter	**Field Size (Type)**	**Description**
PSDMCT	12 (A)	Contact number
PSDMCS	3,0 (S)	Contract supplement
PSSEQ	6,0 (S)	Sequence
PSSRC1	12 (A)	Source 1
PSSRT1	2 (A)	Source type 1
PSDTO	1 (A)	Due To to be used for the contract search/edit

The next three fields represent the item, quantity (with UOM) to be committed.

Parameter	**Field Size (Type)**	**Description**
PSITM	8,0 (S)	Item (short format)
PSQTY1	15,0 (S)	Quantity to be used for contract search/edit
PSUOM	2 (A)	Unit of measure of QTY1

The next four fields represent the order line keys to be used if a committed quantity needs to be added or reversed.

Parameter	**Field Size (Type)**	**Description**
PSDOCO	8,0 (S)	Order number
PSDCT	2 (A)	Order type
PSKCOO	5 (A)	Order company
PSLNID	3,0 (S)	Line number

P38111	365 (A)	Data structure that contains the fields to be used when creating agreement transaction ledger records. Note that this parameter is not passed when only quantity commitment needs to be performed. If it is passed, it will be loaded with the appropriate values by the calling program.

Conversion Programs

P38010AX - Agreement Management file conversion (New)

P49110X - Actual Trip Detail Conversion (New)

Purpose of Program

The Actual Trip Detail Conversion program is a DREAM Writer program that will create Actual Trip Detail (F49110) records from the Trip Detail File (F4911) for trips that have not been delivery-confirmed.

Processing Options

Enter the load confirmation trip status.

Unit of Measure Conversion Server

The UOM conversion server is used to retrieve conversion factors from the primary unit of measure for an item. It can also be used to convert any given unit of measure to any other unit of measure, as long as both units are in either the Item Specific conversion table (F41002) or the Standard conversion table (F41003).

Although there were several changes internal to the UOM conversion server, only two could affect other programs:

> There is a field on the Item Master file (F4101) called Temporary Flash Message (TFLA). This field was added in A7.1 but not used. It will now be used to indicate whether an item will have item-specific units of measure. A ‘1’ indicates only standard units of measure should be searched when calling the server. The field will now be called Standard UOM Conversion.
>
> This modification could affect program testing because the flag may be indicating to bypass item-specific units of measure.

For integrity reasons, this may become a system-maintained field in the future.

There is a set of flags in the parameters that are used to determine which conversion factors are required. These parameters already existed, but were never implemented. Every time the server was called, all conversions were performed.

The flags now can be used to specify which conversions should be performed by moving a ‘1’ to the appropriate flag(s). The exception to this is when all flags are blank, which will still perform all conversions. This exception will allow us to implement the new flags when convenient, rather than performing a mass effort for all calling programs. However, even without the modification for the flags, the UOM conversion server will still perform its function quicker.

The following is a table of the conversion factors and/or quantities that are loaded when a particular input flag is set:

Input Flag	**Flag Description**	**Conv Factor**	**Qty**
#0PQT	#UOM (from) to #UOM1 (to)	#CNV	#QTY
#0SQT	Secondary		#SQT
#0SQT	Secondary		#SQT
#0UQT	Purchasing		#UQT
#0RQT	Pricing		#RQT
#0HQT	Shipping		#HQT
#0LBS	Weight (Pounds)		#LBS
#0CLB	Weight (Hundred Pounds)		#CLB
#0VL1	Volume 1 (Dry)		#VL1
#0CNVQ	#UOM (from) to Primary UOM	#CNVQ	#PQT
#0CNVP	#UOM1 (to) to Primary UOM	#CNVP	

The following parameters are output from the server:

Output Field	**Field Description**
#QTY	Quantity in #UOM1 (to) UOM
#PQT	Quantity in Primary UOM
#SQT	Quantity in Secondary UOM
#UAT	Quantity in Purchasing UOM
#RQT	Quantity in Pricing UOM
#HQT	Quantity in Shipping UOM
#LBS	Weight (Pounds)
#CLB	Weight (Hundred Pounds)
#VL1	Volume 1 (Dry)
#CNV	Conversion Factor Used from #UOM (from) to #UOM1 (to)
#CNVQ	Conversion Factor Used from #UOM (from) to Primary UOM

Here is an example of a call to convert one unit of measure (SFUM) to another ($UOM):

```
CSR                     MOVE LFIVI     #IVI
CSR                     MOVE SFUM      #UOM
CSR                     MOVE $UOM      #UOM1
CSR                     Z-ADD$QTY      #QTY
CSR                     MOVE $MCU      #MCU
CSR                     MOVE *BLANKS   #ERTST
CSR                     MOVE '1'       #0PQT
C*
CSR                     CALL 'X41002  '             81
C*                      ---- ----------
CSR                     PARM           DS4101
CSR                     PARM           DS@UOM
C*
CSR          #ERTST     IFNE '1'
CSR                     Z-ADD#QTY      $QTY
CSR          $XQTY      MULT #CNV      $YQTY
CSR                     END
```

Here is the same call that also converts the quantity ($QTY) to the purchasing (#UQT) and shipping (#HQT) units of measure:

```
CSR                     MOVE LFIVI     #IVI
CSR                     MOVE SFUM      #UOM
CSR                     MOVE $UOM      #UOM1
CSR                     Z-ADD$QTY      #QTY
CSR                     MOVE $MCU      #MCU
CSR                     MOVE *BLANKS   #ERTST
CSR                     MOVE '1'       #0PQT
CSR                     MOVE '1'       #0UQT
CSR                     MOVE '1'       #0HQT
*
CSR                     CALL 'X41002  '             81
C*                      ---- ----------
```

```
CSR                   PARM          DS4101
CSR                   PARM          DS@UOM
C*
CSR         #ERTST    IFNE '1'
CSR                   Z-ADD#QTY     $QTY
CSR                   Z-ADD#UQT     $PURQ
CSR                   Z-ADD#HQTY    $SHIPQ
CSR                   END
```

Manufacturing System Changes

This chapter describes changes made to the manufacturing system.

Configurator Enhancements

Modularization

The existing processing for the configurator was broken into manageable units based on functionality. This structural change was necessary both to simplify implementing current and future enhancements as well as to provide a means of calling certain parts of the configurator through batch programs. The programs that were modularized are Configured Item Specifications (P3294), formerly known as ATO Coded Specification Entry, and the Sales Order Detail Server (X3294). What follows are descriptions of the configured Item Specifications program and the Sales Order Detail Server as they currently exist and descriptions of new programs created.

P3294 - Configured Item Specifications

The Configured Item Specifications program was changed to remove all rule processing. The program includes the calling of the Configurator window, the handling of function keys attached to the window, and the simple edits derived from entering values by using Configured Item Segments (P3291). Even these edits were placed in a copy module executed by the Configured Item Specifications program. The copy module is C3294, Configured Item Segment Editing. This module will process the edits for the UDC tables, numeric yes/no fields, and upper and lower segment values.

The more complicated rule processing that is handled by the configurator is now in Process Assembly Inclusion Rules (P32943). This program will handle both Cross Segment Editing and Assembly Inclusion Rules. Process Assembly Inclusion Rules will be called interactively by Configured Item Specifications and in batch mode by Process Work Orders, (P31410). Interactively, ‘C’ rules or calculated segments will be processed along with ‘P’ rules. In batch mode, ‘Q’ and ‘R’ rules will be processed.

X3294 - Sales Order Detail Server

Prior to the A7.3 enhancement, the Sales Order Detail Server was called once for each sales line item that would be written by the configurator. P3294 and X3294 were changed so that the Sales Order Server will only be called once to update a configured item. In order to implement this change, the structure of X3294 was modified. Rather than being organized to process one sales line at a time, the program will now loop through the existing logic a number of times equal to the number of records read from the sales detail user index. Following the write of the detail records, the parent record will be written.

The Detail Server is called interactively by Configured Item Specifications (P3294) and in batch mode by Process Work Orders (P31410). When called interactively, the program functions the same as it did prior to the enhancement. All updates that are necessary for a configured sales order are completed. When called from Process Work Orders (P31410), the server is called to write parts lists and create routings. Routings will be created as they were in the past, in the called server Create Configured Routings (X3112).

Rule Processing in Process Work Orders (P31410)

In order to improve the interactive performance of the configurator as well as provide consistency for processing parts of the work order, routing rules or 'R' type rules will now be interpreted during the run of Process Work Orders rather than interactively. This modification causes parts list and routings to be created at the same time. To accommodate clients that want routings and parts written soon after the creation of the sales order and work order, a subsystem processing mode is now offered. You can start the subsystem by accessing the Subsystem Control programs. These programs were added to the Configurator Advances Functions menu. To activate the subsystem, choose Start Subsystem from the menu. After this step completes, a field with configurator constants is set to process parts lists and routings using the subsystem.

In addition to processing 'R' and 'P' rule types, Process Work Orders (P31410) will also process the new 'Q' rules. This rule type was also implemented to improve interactive performance. Parts that do not need to be included as a line item on the sales order can be 'Q' rather than 'P' rules. The part will still appear on the parts list but will not be processed interactively and placed on the sales order. 'Q' rules are processed only interactively when the kit pricing method chosen is price by components. To calculate an accurate price or cost for the configured item, 'Q' rules are interpreted. However, the 'Q' parts rules are only used as far as determining price or cost, and when the Configurator Constant option 'Cost Quotes' is set to 'Y'. Updates to the sales order will not take place.

Storing Configured Strings

Two new files were created to facilitate the implementation of stocking configured items. The files are:

Configured String Master (F32941)

A new file was created that contains a “string” of information for each configuration. This string will be composed of segment values for all levels of the configuration, including respective configured items for multi-level configured items. Each occurrence in this file will be unique by configured string identifier per item.

Configured String Segments (F32942)

This file contains the same information as the Configured String Master file except that each segment and its respective value will be stored in a separate record.

In addition to being used for keeping track of the different configurations that can be stocked, these files were also found to be useful when interpreting rules from Process Work Orders (P31410). Assembly Inclusion Rules can be based on values input at any level of a configuration. Prior to the creation of these files, the configured string was pulled apart, separated, and each piece of the string was read until the necessary portion was located. Through the use of the string files, batch processing can be accelerated by finding a value with a single read to the Configured String Detail.

The configured string within Configured String History (F3294) has been increased from 360 bytes to 500 bytes. This represents the maximum length of a string for each configured component, at each level, as well as the configured parent. The Configured String Master (F32941) has a 1000-byte field used to store a concatenation of configured strings at all levels of a configuration. The 1000-byte field is used to search for a matching string to reuse configured string ID numbers. Although the optimal situation would be a concatenated string that is less than 1000 bytes, the F32941 file also contains a sequence number used for overflow. Processing overflow records will not perform as quickly as the direct approach of searching on the first 1000 bytes.

Sequence Number

Sequence number is used by configurator to determine the level at which the configured item is being processed. This field was increased in size from 15 to 23 bytes. Prior to this change, configurations were limited to five levels. Each level could possibly have 99 sales order lines. Because of the increased size of this field, clients can now process as many as 10 levels for a configured item.

External Program Parameters

The parameter list used when an external program is called was changed for version A7.3. Prior to this release, the call using the external program field that is available in Assembly Inclusion Rules was coded as follows:

```
CSR                    CALL KYPID
C*                     ---- -----
CSR                    PARM           PSITM   80
CSR                    PARM           PSMCU  12
CSR                    PARM           PSAN8E  80
CSR                    PARM           PSEXVR 30
C*
CSR                    MOVEAPSEXVR    @IC
```

Today the parameter list appears as:

```
C*
CSR                    CALL KYEPGM
C*                     ---- ------
CSR                    PARM           PSITM   80
CSR                    PARM           PSMCU  12
CSR                    PARM           DSKITP               Sales info
CSR                    PARM           PSEXVR 30
C*
```

The difference between the two calls is that the third parameter is a data structure that holds all of the information passed in by Sales Order Entry. This gives clients access to the sales order header information, detail line information, and processing option values that are all passed to Configured Item Specifications (P3294) from Sales Order Entry (P4211).

Configurator Constants

Configurator Constants (F3209)

A new file was added that contains fields to control configurator processing at the branch/plant level. Some of the fields were removed from the Inventory Constants display file, other fields replaced existing Sales Order Entry processing options, and several new fields were added. The contents of the file are as follows:

Branch/Plant (VDMCU)

Segment Delimiter (VDSECD) - Defaults to '/'

Method of Processing (VDAPLR) - Defaults to '1'; batch

Check Availability (VDCAVL) - Defaults to 'Y'

In Stock Line Type (VDSVL) - Defaults to 'S'

Sales Quote Document Type (VDDCTO) - No default; UDC table

Cost Sales Quotes (VDCCSQ) - Defaults to 'Y'

Work Order Change Status (VDWOCS) - No default

Display Calculated Segments (VDCSEG) - Defaults to 'N'

Configured Model Text

Configured Model Text (F3214)

This file was created as a tag file to Sales Order/Purchasing text file F4314. The Configured Model Text file contains the sequence number for all levels of the configuration. When text is placed on a work order, this file is used to quickly determine which text lines match up with specific levels of the configuration.

Configurator Table Processing

A new feature was created to enable rule processing to perform a lookup from a pre-defined table based on segment values selected in the configuration process. A field was added to Assembly Inclusion Rules to provide for entry of the name of the table to be accessed. This new feature will include programs to create and maintain tables, as well as a retrieval program.

To provide enough flexibility in table usage, tables may be one of three types: prices, part numbers, or calculated segments. Prices are always numeric, part numbers are always alphanumeric, and calculated segments may be either, as defined in the Configured Item Segments. Numeric calculated segment values will be rounded to the number of decimals specified in the segment's definition. Calculated segment and part tables will have the capability of returning multiple values. When defining tables, users may specify up to ten dimensions. Files created for rules-table processing are as follows:

Rules Table Definition (F3281)

A file was added for defining tables including description, table type, number of segments required to look up a value, and whether the lookup will return multiple values.

Configured Item/Rules Table Cross-Reference (F3282)

A file was added to define which segment values to use in table lookups for each configured item. Segments may be defined at different levels of the configuration. For tables with multiple values, the number of values are defined in this file.

Rules Table Value Definition (F32821)

A file was added for tables which will return multiple values to calculated segments. This file holds the segments that will be populated with the returned values. Segments can be populated at any level of the configuration.

Table Detail (F3283)

A file was added to store the actual table values for each combination of segment values defined for the table.

Forecast Consumption Logic

A new planning rule (Rule H) was added to allow the use of forecast consumption logic. This logic will prevent the problem of demand being duplicated because a sales order does not fall on the same date as a forecast. Forecast consumption logic allows MRP to "consume" the forecast with sales orders and shipped quantities within user-defined forecast consumption periods.

Forecast consumption periods are defined by a new program (P3405) and are stored in new file F3405.

When forecast consumption logic is being used, changes are made to the bucket date array depending on the number of past due periods requested in the MRP (P3482/P3483). If two past-due periods are requested, the second element in the Bucket Date Array references the beginning date of the current Forecast Consumption Period (FCP). The first element in the Bucket Date Array references the date prior to the ending date of the previous FCP. If only one past-due period is requested, the first element of the Bucket Date array references the beginning date of the current FCP. If no past-due buckets are requested, the first element of the Bucket Date array references the generation date.

Server XF3462

This server is used to provide access to the forecast shipment summary file. This server is currently used by P4205 (Shipment Confirmation), P42800 (Sales Update), and P49700 (Cycle Billing).

An example of the use of this server follows:

```
C*    Write/update the Forecast Summary record for planning
C*     if planning time fence rule = H (Consumption Periods)
C*
CSR         IBMPSP    IFEQ 'H'
C*
C*    Check Doc Type against UDC 40/CF
C*
CSR                   CLEARI0005U
CSR                   MOVEL'40'      #USY
CSR                   MOVEL'CF'      #URT
CSR                   MOVE SDDCTO    #UKY
CSR                   CALL 'X0005'                83
C*                    ---- -------
CSR                   PARM           I0005U
C*
CSR         #UERR     IFEQ '0'
C*
C*
CSR                   Z-ADDSDITM     MBITM
CSR                   MOVE SDMCU     MBMCU
CSR                   Z-ADDSDPDDJ    MBPDDJ
C*
CSR                   MOVEL'A73   '  @@FMT
CSR                   MOVE 'K'       @@ACCS
CSR                   MOVEL'MBKY01'  @@KLST
CSR                   MOVE 'Y'       @@LOCK
CSR                   MOVEL'CHAIN'   @@OPER
CSR                   Z-ADD3         @@KNUM
CSR                   CALL 'XF3462'
C*                    ---- --------
CSR                   PARM           PS@@1
CSR                   PARM           I3462
C*
CSR         @@IOR     IFNE 'ERR'
C*
CSR         @@IOR     IFEQ 'NF '
C*
CSR                   Z-ADDSDITM     MBITM
CSR                   MOVE SDLITM    MBLITM
CSR                   MOVE SDAITM    MBAITM
CSR                   MOVE SDMCU     MBMCU
CSR                   MOVE SDCO      MBCO
CSR                   Z-ADD$WK150    MBUORG
CSR                   Z-ADDSDPDDJ    MBPDDJ
CSR                   MOVE ##JOBN    MBJOBN
CSR                     MOVE ##PROG    MBPID
CSR                     Z-ADD$#UPMJ    MBUPMJ
CSR                     Z-ADD$TDAY     MBTDAY
CSR                     Z-ADD$TDAY     MBTDAY
CSR                     MOVE ##USER    MBUSER
C*
CSR                     MOVEL'WRITE'   @@OPER
CSR                     MOVEL'MBKY01'  @@KLST
CSR                     CALL 'XF3462  '
C*                      ---- -------
CSR                     PARM           PS@@1
CSR                     PARM           I3462
C*
CSR                     ELSE
C*
CSR                     ADD  $WK150    MBUORG
CSR                     MOVE ##JOBN    MBJOBN
CSR                     MOVE ##PROG    MBPID
CSR                     MOVE $#UPMJ    MBUPMJ
CSR                     MOVE $TDAY     MBTDAY
CSR                     MOVE $TDAY     MBTDAY
CSR                     MOVE ##USER    MBUSER
C*
CSR                     MOVEL'A73     '@@FMT
CSR                     MOVE 'K'       @@ACCS
CSR                     MOVEL'UPDAT'   @@OPER
CSR                     MOVEL'MBKY01'  @@KLST
CSR                     Z-ADD3         @@KNUM
CSR                     CALL 'XF3462  '
C*                      ---- -------
CSR                     PARM           PS@@1
CSR                     PARM           I3462
C*
CSR                     ENDIF
C*
CSR                     ENDIF
```

Summarized Manufacturing J/Es

Manufacturing processing has been enhanced to provide the following:

Ability to summarize Material Issue J/Es *within* a work order

Ability to summarize J/Es *across* work orders

Option to print a summarized Accounting Transaction Report

Summarization is defined by way of the processing options, and impacts P31802, P31804, and P31842.

A new file (F3108) has been created to store the link between summarized work order number and the individual work orders that were summarized. A corresponding inquiry screen was created to view the new file.

Multiple Supplier Split Percentages

Supplier Release Scheduling (SRS) has been enhanced to allow generation of multiple supplier schedules for a single item. Additionally, a pre-determined split percentage between suppliers can be defined.

Previously, a blanket order was in effect through its expiration (request) date. It is now further defined as being effective from the order date through the expiration date.

A new file (F43211) has been created to store the respective supplier percentages and corresponding effectively dates.

If an item has been split among suppliers, the SRS Schedule Revision program (P34301) was modified to invoke a window to allow selection of the desired supplier/blanket order.

Warehouse Management Picking Interface

The Warehouse Management Picking Interface project objectives are:

Provide clients with the flexibility to decide whether or not the pick requests will be generated at the time a parts list is created

Enhance manufacturing parts list generation programs to call the WM Pick Request server

Add a check for availability of parts attached to work center locations

Update the Pick Request, Pick Confirmation, and Request Cancellation programs to recognize requests originating in manufacturing

Change warehouse programs to update the appropriate fields in the Parts List file rather than the Sales Order Detail file when a request originates in manufacturing

There were no database changes required to implement the interface. However, new codes were added to the Material Status UDC table (31/MS). In addition, the glossary for the data item OCDE was changed to include 'WO' as a new origin code type.

Engineering Change Order Enhancements

The enhancement to the existing ECO functionality consisted primarily of the following new features:

Ability to create work orders against prior ECO revision levels

Provide automatic assignment of the next revision level during ECO entry through control of a processing option

Allow updates of all single-level parent revision levels whenever a component within its structure changes

Allow approval routing assignment for an individual ECO order

To implement the enhancement, the following technical changes were made:

User Defined Codes (F0005)

A new entry was added to the UDC table to contain pre-assigned revision levels. This table will be used during ECO entry to determine the next revision level to assign.

ECO Parts List (F3013)

To facilitate bill of material maintenance, Issue Type Code (ITC) was added.

Last Drawing Revision Level (RVNO), which was not utilized previously, was included on ECO Parts List Maintenance (P3013).

Order Approval Routing Logical (F4818LB)

To provide approval routing at the ECO level, a new logical file was created to sequence the records by ECO order, then approval group. This permits releasing different routing groups during the approval process.

ECO Parts List Logical (F3013LE)

To enable Revision window to provide Skip To functionality, a logical view of the ECO parts list by Parent Item and Swap to Revision was added.

ECO Parts List Logical (F3013LF)

To enable Revision window to provide Skip To functionality, a logical view of the ECO parts list by Item Number and To Revision was added.

Human Resources and Payroll Systems

This section provides database changes for the human resources and payroll systems.

Database Changes

Several database changes have been made to the Payroll and HRM systems to accomodate enhancements made for public services clients. The areas affected the most by these changes involve the ability for employees to have more than one position/job, and enhancements to position budgeting and control.

In addition to public services, enhancements have been made to the HRM and Payroll systems for Canadian Employment Equity Federal requirements.

Files Common to Payroll and HR - System 05

F060116 - Employee Master - YA

Description	Size	Type	Data Item	Status	Conversion Value
Pay Grade Step	4	A	PGRS	Add	Blank
Salary Forecast Change Date	6	P	DTSF	Add	Zero
Num Pay Periods/Year	6	P	SMOY	Add	Zero
Std Days/Year	5	S	SDYY	Add	Zero
Hours - Standard Per Day	5	S	STDD	Change	Change from Size 5, File Decimals 1, to Size 5, File Decimals 0, Display Decimals 2.
Hours - Standard Per Payperiod	5	S	STDH	Change	Change from Size 5, File Decimals 1, to Size 5, File Decimals 0, Display Decimals 2.

F060118 - Employee Multiple Job - YE - New File

Description	Size	Type	Data Item	Status	Conversion Value
Employee Address No	8	S	AN8	Add	From F060116
Home Business Unit	12	A	HMCU	Add	From F060116
Job Type	6	A	JBCD	Add	From F060116
Job Step	4	A	JBST	Add	From F060116

Position ID	8	A	POS	Add	From F060116
Primary Job Flag	1	A	EJPF	Add	‘P’
Pay Starts Date	6	S	PSDT	Add	From F060116
Pay Stops Date	6	S	PTDT	Add	From F060116
Union Code	6	A	UN	Add	From F060116
Annual Salary	11	P	SAL	Add	From F060116
Hourly Rate	11	P	PHRT	Add	From F060116
Auto Pay Type	3	A	ATPY	Add	From F060116
Locality	8	A	SLOC	Add	From F060116
Shift Code	1	A	SHFT	Add	From F060116
Worker’s Comp Code	4	A	WCMP	Add	From F060116
Worker’s Comp Class	1	A	WET	Add	From F060116
Pay Grade	6	A	PGRD	Add	From F060116
Pay Step	4	A	PGRS	Add	From F060116
FLSA	1	A	FLSA	Add	From F060116
EEO Category Code	3	A	EEOJ	Add	From F060116
Supervisor	8	A	ANPA	Add	From F060116
Pay Class	1	A	SALY	Add	From F060116
Employment Status	1	A	EST	Add	From F060116
Compa Ratio	3	A	CMPA	Add	From F060116
Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Next Review Date	6	S	NRVW	Add	From F060116
Type Review	1	A	TINC	Add	From F060116
10 Category Codes	3	A	P001-10	Add	From F060116
10 User Defined Dates	6	S	ED01-10	Add	From F060116
FTE	9	P	FTE	Add	From F060116
Number of Periods Paid per Year	5	P	SMOY	Add	From F060116
Standard Hours Worked per Year	7	P	IH	Add	From F060116
Standard Hours Worked per Day	5	S	STDD	Add	From F060116
Standard Days Worked per Year	5	S	SDYY	Add	From F060116
Pay on Standard Units	5	S	STDH	Add	From F060116

Salary Forecast Change Date	6	S	DTSF	Add	From F060116
Change Reason	3	A	TRS	Add	From F060116
Effective Date	6	S	EFTO	Add	From F060116
User ID	10	A	USER	Add	From F060116
Program ID	10	A	PID	Add	From F060116
Date - Updated	6	S	UPMJ	Add	From F060116
Work Station ID	10	A	JOBN	Add	From F060116

F060119 - Employee Multiple Job History - YE - New File

Same as F060118 except file is keyed as non-unique.

Conversion - Copy records from F060118 into F060119.

F06106 - Employee Pay Instructions - YM

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Position ID	8	A	POS	Add	From F060116
Home Business Unit	12	A	HMCU	Add	From F060116

F06116 - Employee Transaction Detail - YM

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Position ID	8	A	POS	Add	From F060116

F0618 - Employee Transaction Detail History - YT

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Position ID	8	A	POS	Add	Blank

F0618RT - Payroll Transaction History Workfile - YT

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Position ID	8	A	POS	Add	None

F0618WF - Payroll Transaction History Workfile - YT

Description	Size	Type	Data Item	Status	Conversion Value
Position ID	8	A	POS	Add	None

F0618WK - Payroll Transaction History Workfile - QW

Description	Size	Type	Data Item	Status	Conversion Value
Position ID	8	A	POS	Add	None

F08001 - Job Master - JM

Description	Size	Type	Data Item	Status	Conversion Value
Pay Grade Step	4	A	PGRS	Add	Blank
Status Change Reason	3	A	STCR	Add	Blank
Canadian NOC Code	4	A	CENC	Add	Blank

F08040 - HR Constants - JT

Description	Size	Type	Data Item	Status	Conversion Value
Use Assignment Window	1	A	YORN	Add	‘N’
Pay Rate Source	1	A	PRSR	Add	‘3’
Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Step Progression Rate Source	1	A	SPRS	Add	‘2’
Flag - Position Budget Edit 1	1	A	PBF1	Add	Blank
Flag - Position Budget Edit 2	1	A	PBF2	Add	Blank
Flag - Position Budget Edit 3	1	A	PBF3	Add	Blank
Flag - Position Budget Edit 4	1	A	PBF4	Add	Blank

Flag - Pay Range Edit	1	A	PRF1	Add	Blank
Salary Increases in Projections	1	A	SIP	Add	‘N’
Salary Default Source	1	A	SDFS	Add	Blank
Salary Display	1	A	SALD	Add	‘A’

F08101 - Position Master - HP

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Position Status	1	A	PSST	Add	Blank
Budget Status Date	6	S	PBSD	Add	Zero
Position Status Date	6	S	PSSD	Add	Zero
Locality	8	A	SLOC	Add	Blank
Position Budget Amount - Effective	15	P	EFFB	Add	Calculate from CURB using Percentage of Year server - X081011
Position Budget Hours - Effective	11	P	EFHR	Add	Calculate from HRB using Percentage of Year server - X081011
Position Budget FTE - Effective	9	P	FTEE	Add	Calculate from BFTE using Percentage of Year server - X081011
Position Budget Headcount	9	P	HDCT	Add	Zero

F081012 - Position Account - HC - New File

Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
FY	2	S	FY	Add	
FY	2	S	FY	Add	
Home Business Unit	12	A	HMCU	Add	
Position ID	8	A	POS	Add	
Subledger	8	A	SBL	Add	
Subledger Type	1	A	SBLT	Add	

Account Number	8	A	AID	Add
Percentage	15	P	PAPC	Add

F08102 - Requisition Master - HN

Description	Size	Type	Data Item	Status	Conversion Value
Headcount	9	P	HDCT	Add	Zero
Pay Grade Step	4	A	PGRS	Add	Blank
Filled By Date	6	S	FBDT	Add	Zero

F08105 - Requisition Activity - HM

Description	Size	Type	Data Item	Status	Conversion Value
Candidate Requisition Status	2	A	CRST	Add	Blank
Candidate Requisition Status Date	6	s	CRSD	Add	Zero

F082001 - Pay Grade/Salary Range Table - HS

Description	Size	Type	Data Item	Status	Conversion Value
Pay Grade Step	4	A	PGRS	Add	Blank
Pay Grade Step Rate	15	P	PGSR	Add	Zero
Union Code	6	A	UN	Add	Blank
Next Pay Grade	6	A	NPGD	Add	Blank
Next Pay Grade Step	4	A	NPGS	Add	Blank
Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
Std Hrs/Day	5	S	STDD	Add	Zero
Std Days/Yr	5	S	SDYY	Add	Zero

F08201 - Salary Review Work File- HR

Description	Size	Type	Data Item	Status	Conversion Value

Union Code	6	A	UN	Add	Blank

HR Files - System 08

T082003 - Grade Step Progression Workfile- YW - New File

Description	Size	Type	Data Item	Status	Conversion Value
Review Group ID	6	A	SRVW	Add	
Employee Address No	8	S	AN8	Add	
Home Business Unit	12	A	HMCU	Add	
Job Type	6	A	JBCD	Add	
Job Step	4	A	JBST	Add	
Primary Job Flag	1	A	EJPF	Add	
Pay Stops Date	6	S	PTDT	Add	
Union Code	6	A	UN	Add	
Locality	8	A	SLOC	Add	
Pay Grade	6	A	PGRD	Add	
Pay Step	4	A	PGRS	Add	
Annual Salary	11	P	SAL	Add	
Hourly Rate	11	P	PHRT	Add	
FTE	9	P	FTE	Add	
Review Group Creation Date	6	S	RGCD	Add	
New Annual Salary	11	P	SALN	Add	
New Hourly Rate	11	P	NHRT	Add	
New Pay Stops Date	6	S	NTDT	Add	
Description	**Size**	**Type**	**Data Item**	**Status**	**Conversion Value**
New Pay Starts Date	6	S	NSDT	Add	
New Pay Grade	6	A	NPGD	Add	
New Pay Grade Step	4	A	NPGS	Add	
Name	40	O	ALPH	Add	

Review Group Desc	30	O	DL01	Add

Architecture/Engineering/Construction Systems

This section provides database changes for the following systems:

Fixed assets

Property management

Work orders

Contract management

Homebuilder management

Service billing

Contract billing

Job cost

Equipment management

Database Changes

This chapter lists database changes for fixed assets, property management, work orders, contract management, homebuilder management, service billing, contract billing, job cost, and equipment management systems.

Fixed Assets

New Files

File	File Name	File Contents	Prefix
F1201JD	Item Master / Item Balance Joined Logical File	This is a joined logical file over the Item Master File (F1201) and the Item Balance File (F1202).	FA, FL

F12021	Item Balance File Prior Year A/D Balance File	This file is a tag file to the F1202 that is used to carry the portion of a current year balance that pertains to a prior year balance that has arisen as the result of a transfer or split. Particularly where a subledger is used and a transfer occurs, the current year portion transferred to a new subledger reflects the entire balance. This allows tracking of the prior year component for depreciation calculation purposes. The file is OPTIONAL, pending existence of fixed assets.	L1
F12022	Item Balance File Forecast File	This file is a tag file to the F1202 used (at present) for calculating future year depreciation for incorporation into depreciation projections. The file is OPTIONAL, pending existence of fixed assets.	L2

File	**File Name**	**File Contents**	**Prefix**
F12851	Depreciation Rule Header (User Defined Depreciation)	This file contains the elements necessary to define a depreciation rule, and the conventions and rules that are applicable to all years that the rule is in effect. This file will contain J.D. Edwards rules and may also contain rules constructed by a client. J.D. Edwards rules are characterized by NUMERIC depreciation methods; client rules are characterized by ALPHA depreciation methods. The file is OPTIONAL, pending existence of fixed assets. If user-defined depreciation is not used, the file may be cleared.	LU

File	File Name	File Contents	Prefix
F12852	Annual Depreciation Rule	This file contains references to all of the formula elements necessary to calculate depreciation for a year or range of years for a particular rule. The rule is subdivisible to the “Placed In Service” month if necessary. The file is a subsidiary to the Depreciation Rule Header (F12851), and will share the key to that file, with the additional key of life year and placed in service period. This file will contain J.D. Edwards rules and may also contain rules constructed by a client. J.D. Edwards rules are characterized by NUMERIC depreciation methods; client rules are characterized by ALPHA depreciation methods. The file is OPTIONAL, pending existence of fixed assets. If user-defined depreciation is not used, the file may be deleted.	LV
F12853	Depreciation Formulas	This file contains the formula rules that are referenced by the annual depreciation rules for various elements of the depreciation equation. They are keyed by a formula ID This file will contain J.D. Edwards rules and may also contain rules constructed by a client. J.D. Edwards rules are characterized by a NUMERIC formula ID; client rules are characterized by ALPHA IDs. The file is OPTIONAL, pending existence of fixed assets. If user-defined depreciation is not used, the file may be deleted.	LW
F12854	Depreciation Spread Patterns	This file contains a series of percentages corresponding to fiscal periods. The percentages are applied to an annual depreciation amount calculated by an annual rule resulting in the depreciation allocated to individual fiscal periods. The file is OPTIONAL, pending existence of fixed assets. If user-defined depreciation is not used, the file may be deleted.	LZ

New Logical Files

File	Key Fields
F12851LA	ADM, ADLM, ITAC, DIR1, DTFR, EFTB
F12852LA	ADM, ADLM, ITAC, DIR1, DTFR, EFTB, LYFR, PSIP, SPCN=‘1’
F12852LB	ADM, ADLM, ITAC, DIR1, DTFR, EFTB, LYFR, PSIP, SPCN=‘2’
F12853LA	DC20

Other Database Changes

File	File Name	File Changes
F1202	Item Balance File	Balance Character Code - The Balance Character Code was implemented with A7.3. This code identifies the “Character” of the balance as to whether it is a cost, an accumulated depreciation, a depreciation expense, or a disposal account dependant upon automatic accounting instructions.

It is IMPORTANT that if you use F1202, you must initialize this field to BLANK if the field is NOT one of the above categories. Consult the glossary for the element CHCD for valid values.

Property Management

No database changes have been made to the property management product between A7.1 and A7.3.

Work Orders

No database changes have been made to the work order product between A7.1 and A7.3.

Contract Management

No database changes have been made to the contract management product between A7.1 and A7.3.

Homebuilder Management

The homebuilder management product is new for A7.3. For database information, refer to the *Homebuilder Reference Guide*.

Service Billing

New Physical Files

File	File Name	File Contents
F48221	Service Billing Retention Release Cross-Reference File	This file is designed to provide a cross-reference capability between the invoice created to release retention and the invoices on which the retention was originally calculated.

Fields Added

File	File Name	Fields Added	
F4812	Service Billing Workfile	RSBF	Restatement Basis Flag
		ITM	Item Number
		PAID	Price Amount
		SUBA	Alternate Cost Code
		ERDB	Exchange Rate Date Basis
		BCTK	Batch Control Key
		IDGJ	Invoice G/L Date
		FEA	Foreign Price
F4812H	Service Billing Workfile History	RSBF	Restatement Basis Flag
		ITM	Item Number
		PAID	Price Amount
		SUBA	Alternate Cost Code
		ERDB	Exchange Rate Date Basis
		BCTK	Batch Control Key
		IDGJ	Invoice G/L Date
		FEA	Foreign Price

File	File Name	Fields Added

F4822	Invoice Summary File	SMRP	Stored Materials Retainage - Prior Amount
		SMPF	Stored Materials Retainage - Prior Amount - Foreign
		RTHP	Restated Billing Amount
		RSBF	Restatement Basis Flag
		PTAM	Prior Tax Amount
		FPTA	Prior Tax Amount - Foreign
		RGLC	Retention Offset
		PRET	Retention Percent
		FUP	Unit Price - Foreign
		SBL	Subledger
		SBLT	Subledger Type
F48011	Batch Control File	BCTK	Batch Control Key
		BCTC	Batch Control
F48091	Service Billing System Contstants	ERDB	Exchange Rate Date Basis
		MBGC	Multiple Batch Generation Control
		DTAI	Data Item
		BCT1	Batch Control Flag 1
		BCT2	Batch Control Flag 2
		BCT3	Batch Control Flag 3
F48096	Cost Plus Markup Table	CTF1	Control Flag 1
		CTF2	Control Flag 2
			The position of the Currency Code field (WQCRCD) changed from the first key of this keyed physical file to the fourth key. This will improve performance when this module is fully enabled for multi-currency.

Other Database Changes

File	File Name	File Changes
F48096 F48096LB F48096LC F48096LD F48096LE F48096LF F48096LG	Cost Plus Markup Table	The position of the Currency Code field (WQCRCD) changed from the first key of this keyed physical file to the fourth key. This will improve performance when this module is fully enabled for multi-currency.

File	File Name	File Changes

F4860	Component Table Master	The position of the currency code field (AFCRCD) changed from the first key of this keyed physical file to the second key. This will improve performance when this module is fully enabled for multi-currency.
F4861 F4861LA F4861LB	Component Table Detail	The position of the currency code field (AFCRCD) changed from the first key of this keyed physical file to the second key. This will improve performance when this module is fully enabled for multi-currency.
F4862 F4862LA F4862LB	Component Cross Reference	The position of the currency code field (AFCRCD) changed from the first key of this keyed physical file to the second key. This will improve performance when this module is fully enabled for multi-currency.

Contract Billing

Fields Added

File	File Name	Fields Added
F5201	Contract Master File	RSBF Restatement Basis Flag RNTE Restated Guarantee Maximum Amount RGLC Retainage G/L Offset CRRD Exchange Rate For Guaranteed Maximum Amount ERDB Exchange Rate Date Basis BCTK Batch Control Key PYWP Pay When Paid Flag AI11 Contract Category Code #11 AI12 Contract Category Code #12 AI13 Contract Category Code #13 AI14 Contract Category Code #14 AI15 Contract Category Code #15

F5202	Owner Pay Item Header	RSBF	Restatement Basis Flag
		RGLC	Retainage G/L Offset
		FSOF	Schedule Of Values - Foreign
		FUP	Unit Price - Foreign
		RSOF	Schedule Of Values - Restated
		RUP	Unit Price - Restated
		RNTO	Not To Exceed Amount - Re stated
		RRRA	Unit Price - Restated
		RNTE	Not To Exceed Amount - Line Level - Restated
		CRRD	Exchange Rate for Recurring Amount and Not To Exceed - Line Level
		FNTE	Note To Exceed - Line Level - Foreign
		FNTO	Not To Exceed Amount - Foreign
		FRBA	Recurring Amount - Foreign
		PYWP	Pay When Paid Flag
		ERDB	Exchange Rate Data Basis

Other Database Changes

File	**File Name**	**File Changes**
F5201E	Contract Master File - Compile Purposes Only	Reflect the same changes as F5201
F5202E	Owner Pay Item Header - Compile Purposes Only	Reflect the same changes as F5202

Job Cost

Fields Added

File	**File Name**	**Fields Added**
F5144	Profit Recognition File	VER Version (Historical)

Equipment Management

Fields Added

File	File Name	Fields Added	
F1207 F1207E	Item PM File	UKID	Unique Key ID
		TOLU	Upper Tolerance Limit
		TOLL	Lower Tolerance Limit
		HLDD	Hold Date
		SPHR	Specific Hours
		SPWK	Specific Weeks
		SPMN	Specific Months
		WK	Week Number
		SPDW	Week Day
F1300 F1300E	Equipment Constants	TYPR	Record Type

Fields Changed

File	File Name	Field Changes		Field Size
F1307	Status History File	BEGT	Beginning Time	Size Changed to 6.0
F1307E		ENDT	Ending Time	Size Changed to 6.0

New Files

File	File Name	File Contents	Prefix
F1215	Specification Cross Reference	This new file contains the specification data cross reference information that allows you to completely customize your equipment specification data. For each type of equipment, it defines which fields in the Specification Data file (F1216) are valid, the size of the field, the type of field it is, how it is displayed, and how it is edited.	GV

F1216	Specification Data File	This new file contains the equipment specification data that could be nameplate information about the piece of equipment or specification sheet information from the manufacturer of the piece of equipment. The information in this file is completely user-defined based on the specification cross reference and could include such things as model, number, power requirements, operation instructions, dimensions, and so on.	GZ
F12161	Specification File Edit	This new file contains all of the possible sizes of fields for the Specification Data File, but does not actually contain any data. It is used to edit a particular specification data field against the specified file from the specification cross-reference based on the size of the field defined in the Specification Cross Reference File.	GZ
F1308	Maintenance Loops	This new file contains all of the pieces of equipment that will be part of a maintenance loop defined as one PM task for the "parent" piece of equipment. When the "parent" PM task comes due, all of the other pieces of equipment defined in the maintenance loop will also be attached to the work order. This allows the same PM task to be done to multiple pieces of similar equipment at the same time under one work order instead of one work order for every piece of equipment.	F0

New Joined Files

File	File Name	Prefix
F1201JC - F1201/F1216	Item Master/Specification Data - Joined Logical File	FA/GZ
F1201JE - F1201/F4801	Item Master/Work Order Master - Joined Logical File	FA/WA

New Logical Files

File	File Name	Key Fields
F1207LE	LF - Assigned WO, Item Number	WONA, NUMB
F1207LF	LF - Unique Key ID	UKID
F12161JA	LF - Alpha 21	AA21
F12161JB	LF - Alpha 22	AA22
F12161JC	LF - Alpha 23	AA23
F12161JD	LF - Alpha 24	AA24
F12161JE	LF - Alpha 25	AA25
F12161LA	LF - Alpha 01	AA01
F12161LB	LF - Alpha 02	AA02
F12161LC	LF - Alpha 03	AA03
F12161LD	LF - Alpha 04	AA04
F12161LE	LF - Alpha 05	AA05
F12161LF	LF - Alpha 06	AA06
F12161LG	LF - Alpha 07	AA07
F12161LH	LF - Alpha 08	AA08
F12161LI	LF - Alpha 09	AA09
F12161LJ	LF - Alpha 10	AA10
F12161LK	LF - Alpha 11	AA11
F12161LL	LF - Alpha 12	AA12
File	**File Name**	**Key Fields**
F12161LM	LF - Alpha 13	AA13
F12161LN	LF - Alpha 14	AA14
F12161LO	LF - Alpha 15	AA15
F12161LP	LF - Alpha 16	AA16
F12161LQ	LF - Alpha 17	AA17
F12161LR	LF - Alpha 18	AA18
F12161LS	LF - Alpha 19	AA19
F12161LT	LF - Alpha 20	AA20
F12161LU	LF - Numeric 01	IA01
F12161LV	LF - Numeric 02	IA02
F12161LW	LF - Numeric 03	IA03

F12161LX	LF - Numeric 04	IA04
F12161LY	LF - Numeric 05	IA05
F12161LZ	LF - Numeric 06	IA06
F12161L1	LF - Numeric 07	IA07
F12161L2	LF - Numeric 08	IA08
F12161L3	LF - Numeric 09	IA09
F12161L4	LF - Numeric 10	IA10
F12161L5	LF - Numeric 11	IA11
F12161L6	LF - Numeric 12	IA12
F12161L7	LF - Numeric 13	IA13
F12161L8	LF - Numeric 14	IA14
F12161L9	LF - Numeric 15	IA15
F1308LA	LF - Item Number, Service Type, Parent Number	NUMB, SRVT, AAID
F1308LB	LF - Parent Number, Document Number	AAID, DOCO
F1308LC	LF - Document Number, Parent Number	DOCO, AAID

Programmer's Tools and Considerations

This section provides the following:

- Re-engineering modules and conversion tools
- Performance considerations for programmers
- National language support

Re-Engineering Modules and Conversion Tools

The purpose of the re-engineering modules is to assist in the upgrade of custom source from A5 to A6 or A7. These modules may not work in all cases for a given customer, but the module source gives a good starting point or model that customers can adjust to their individual circumstances.

Customers must review the source for these re-engineering modules to make sure they work at their site. For example, source in a customer library can have a different name than the library in the shipped program. Also, the modules are written specifically for A5 to A6 or A7 conversions, and the customer may be converting from A4.3 to A6 or A7.

Source code is available to be used only as a model, with customization to be performed at customer sites. The modules are not to be used “as is.”

Re-engineering modules are available to perform the following functions:

Change data dictionary access to a server call

Change user defined codes access to a new server call

Change DREAM Writer version length

Change length of install system (SY) and reporting system (SYR)

NOTE: The Data Dictionary Server Space program (X98SRV) is obsolete in version A6.1 and forward. Any references to this program in custom programs MUST be changed to call the new Data Dictionary Server (X9800E).

Recompiling Source Supplied by J.D. Edwards

Members shipped for use in upgrading your custom source code include:

Member	**Description**
J89501	A6/A7 Source Conversion Error Report
P89501X	A6/A7 Source Conversion CL
J89501B	A6/A7 Source Conversion Driver
P89501	A6/A7 Source Conversion RPG
R89501	A6/A7 Source Conversion Error Report

Member	**Description**
J89502	DREAM Writer/Language Preference Conversion CL
P89502	DREAM Writer/Language Preference Conversion RPG
P89502C	DREAM Writer/Language Preference Conversion CLP
P89502E	DREAM Writer/Language Pref Conversion Error Report
P89502O	DREAM Writer/Language Preference Conversion CL
J89510	DDS Fold Area Conversion CL
J89510B	DDS Fold Area Conversion Driver
P89510	DDS Fold Area Conversion
R89510	DDS Fold Area Conversion Error Report

J89511	DDS Fold Area Key Word Conversion CL
J89511B	DDS Fold Area Key Word Conversion Driver
P89511	DDS Fold Area Key Word Conversion
R89510	DDS Fold Area Key Word Conversion Error Report

All members except P89502C must be recompiled before processing. DREAM Writer versions have been provided, (forms P89501, P89510, and P89511).

Specific Functions - What They Do

RPG Changes

Change all CHAINs to F9800 and F0005 to use a server.

F9800 uses X9800E
F0005 uses X0005

Delete file specs for F9800 and F0005 if open for input only, and there are only CHAINs to the file.

Delete key lists for F9800 and F0005 if the file specification is deleted. The comments before the key list are also deleted.

Delete OPEN and CLOSE operations to F9800 and F0005 if the file specification is being deleted.

Change length of system code field in subroutine S998 to four characters.

Change user defined codes server XS0005 to new server X0005.

Change any MOVEs to a system code field to a MOVEL. System code fields are identified as:

xxSY	xxSYR
$SY	$SYR
#SY	#SYR

Print a warning message where hard-coded values are being moved to a system code field.

Print a message that identifies where other “potential” system code fields exist.

Add the value “A6WRN” to the first five positions in the source line for those lines that have a warning message. Scan on this value to identify those areas.

Changes parameter lists with VERS and a length of three characters to a length of ten characters.

Prints suspicious references to fields containing "VER" or "VERS" on the report but does not change the source. This is because indirect references cannot be traced. For example, if a field named $#911 is three characters long, and it is moved into $VERS, the fact that $#911 is three characters long cannot be known.

Change MOVE operations to fields that are xxVERS and length three characters to a length of ten characters, and prints them on the report.

Change a move of a literal '001' to a version field to 'ZJDE0001'. Prints a message on the report to inform you of this change.

Most MOVE operations into a version field should be changed to MOVEL. These are NOT changed, but identified on the report. You need to review these items manually to determine what operation is appropriate.

A user defined code table (89/PG) is available to identify any programs which should not be converted at all.

Change user defined code field names as follows:

DRSY	will be	#USY
DRRT		#URT
DRKY		#UKY
DRDL01		#UDL01
DRDL02		#UDL02

Server fields:

D1@@	will be	#UDL01
D2@@		#UDL02

CL Program Changes

Looks for the following conditions on DCL (declare) statements:

Variable name containing the string VERS

TYPE(*CHAR)
Length of 3

The length is changed to 10 if it meets the criteria.

A new section of code is inserted (for batch jobs only) following the global MONMSG, which will perform the operations necessary to override the QJDEMSG file for language preference. A new field is added in the DCL (declare) section for this function. The Function Use code in the software versions repository is used to determine if the program is batch or interactive.

Exceptions to Specific Functions - What They Don't Do

Change system code fields in the I (input) specifications. An attempt is made to identify them and a message prints on the report if they are found.

Convert UPDAT, DELET, SETLL, or READx statements to F9800 and F0005.

Convert any programs with CAP Status equal to "Y".

Convert any reference to OLD Data Dictionary server XS9800. These are identified on the report.

Convert any reference to Data Dictionary Server Space program X98SRV. These are NOT identified on the report.

Convert moves to user defined code key fields which are outside the section of code doing the chain or call to the server.

The language preference override in CL is not added if a global MONMSG is not found. A warning message is printed.

The "A6WRN" inserted on some lines of source should be blanked out after the code is reviewed.

Specific Functions - Reports

General Messages

CAP Gen Status Invalid-pgm not converted

The CAP status is "Y". It must be changed to "N" before the program can be converted.

** Program Bypassed **

The program is in the user defined code table (89/PG) of programs to be bypassed. Delete it from the table if you want to run the conversions.

Questionable Deletions

Any deleted comment lines are printed here. Verify that they are no longer needed and should be deleted. If OPEN or CLOSE statements existed for F9800 or F0005, these will be deleted and printed here.

Source member not found in "from" library

The member does not exist in the library specified as the “from” library. Correct the software versions repository record, and submit the conversion program again.

F9800 not referenced/File Spec Deleted

The F9800 file was not used in the program. No server replacement was performed.

F9800 open for update/S998 converted

The conversion program changed any CHAINs to F9800 in subroutine S998 ONLY to use the server. The program must be changed manually to access the new data dictionary file.

Non-standard access to F9800/partial conversion

The conversion program changed any CHAINs to F9800 to use the server. However, SETLL, READx, UPDAT, or DELET statements were found and were not changed. The program must be changed manually to access the new data dictionary file.

Call XS9800 must be converted manually

This program is calling an obsolete version of the data dictionary server. Remove these calls, and change to the new X9800E server.

F0005 open for update/file not deleted

The conversion program changed any CHAINs to F0005 to use the server. The program may need to be changed manually to update the new fields in the UDC file.

UDC access changed to server

Message for information only.

Non-standard access to F0005/partial conversion

The conversion program changed CHAINs to F0005 to use the server. However, SETLL, READx, UPDAT, or DELET statements were found and were not changed. The program may need to be changed manually to access a different UDC I/O server.

Hard-Coded Literals to System Code Exist

The program is moving literal values to a system code field. This message is provided for information only. It may or may not require a change to the program. You may scan on A6WRN to find the occurrences.

Incorrect field being moved to UDC key

One of the UDC key fields is being loaded with information too large for the field. Scan on A6WRN to find the source line in error and correct the information.

UDC key fields not loaded

The conversion routine was unable to load the UDC key fields properly. Scan on A6WRN to find the section in question and modify the program to load the key fields.

Questionable system code field

The source line is printed that contains a field that may be a system code field. Verify the line in question and make manual changes as required.

Version Conversion Exceptions Occurred

Questionable DREAM Writer version fields were encountered. These lines are printed on the report. These should be reviewed to determine if manual changes are necesary.

No Global MONMSG

A global MONMSG was not found for this CL program. Therefore, the section of code to perform the override to the QJDEMSG file according to language preference must be added manually. Copy this section of code from a CL that has already been converted.

DREAM Writer Version Issues

Naming Convention Issues - Beginning with the A6.1 release, DREAM Writer versions have a new naming convention:

Version	**Description**
XJDExxxxx	The versions that J.D. Edwards ship as DEMO versions, owned by J.D. Edwards, follow the naming convention of XJDExxxx (where xxxx is a version number). These versions are replaced by merges and do not require ASIs.
ZJDExxxxx	The versions that the client owns for “default” (hard-coded) versions follow the naming convention of ZJDExxxx (where xxxx is a version number). The ZJDE versions are not overlaid by merges and require ASIs.

Specific DREAM Writer Conversion

Version	Description
XJDE	All “DEMO” versions that do not have hard-coded references must be changed to “XJDExxxx” where xxxx is the version number.
ZJDE	Any hard-coded DREAM Writer version references should be changed from the old naming convention to the new convention. You should create ZJDE versions only when a hard-coded reference exists, or a direct reference exists in a program or on a menu (such as a blind DREAM Writer execution from a menu). For example, “001” should be changed to “ZJDE0001”.
Processing Option Text	Processing option **text** references to versions should be changed to the new naming convention. Such references normally need to be converted to a ZJDE version. If the **value** of the processing option is a DREAM Writer version, it must be changed to a ZJDE version number. ASIs are necessary for changes to processing option values.
Concatenation issues	The logic associated with concatenations of a DREAM Writer version and another value should be modified to expect a new length. NOTE: IBM names are limited to 10 characters.

Re-Engineering Modules for New DDS Keywords

A re-engineering module is available to add new keywords to display file DDS. One of the primary functions of these keywords is to enable the program to control the display of the fold area. The use of these keywords eliminates the need for program X96DROP.

X96DROP is an MI program that is called when doing any cursor-sensitive help processing. It is used to determine whether a subfile is folded or truncated. This program does not function properly when the AS/400 system is set to run under IBM security level 40. Therefore, if your site is set at security level 40, you MUST perform this conversion on any custom code for interactive programs. If you are not at security level 40, this conversion is optional.

Source code is available to be used as a model only. We assume that you will customize. They are not to be used “as is.”

What the Conversion Will Do

For subfile screens, adds the following keywords:

SFLFOLD (conditioned on indicator 04)
SFLMODE
SFLCSRRRN

Changes the SFLDROP keyword to be conditioned on indicator condition “N04”.

Adds RTNCSRLOC keyword for all display files.

Changes the RPG for interactive programs as follows:

Cursor Sensitive Help program X96CCF changes to X96CCX

Additional PARM inserted - I00MDE

Writes to subfile screens (control records) changes to set SFLDROP/SFLFOLD indicator.

Financial Systems Source Search Report - P98330C

Purpose

This search program is DREAM Writer driven. It can be used on a program-by-program basis. A report that results from this source scan reveals individual source lines that may need modification, along with messages that indicates possible source modification. The processing options will allow you to:

Specify a library for your source search.

Specify a full source print or only the source lines with warning messages.

What the Source Search Will Do

Scan for A6WRN messages generated from Tech REAM. You should investigate these messages and/or remove them.

Scan for XS9800. You should replace all uses of this server with the new server X9800E.

Scan for data items that have new lengths. You should make adjustments to data items, work fields, and data structures in the program.

Scan for company ‘000’ (hard-coded). You should change hard-coded references to company to five digits (‘00000’).

Scan for externally-defined data structures over files that have file servers. You should remove these data structures and replace them with I/O server file format copy modules.

Scan for files that should go through I/O servers (F0006, F0901, and F0101). You must implement I/O servers for these files.

Scan for X09031. You must change the parameters for this routine if X09031 already exists in your code.

Scan for copy module C0903, which is being replaced by external program X09031.

Scan for calls to X091101. These calls can optionally be changed to a WRITE to the F0911 file.

Scan for cost center being scrubbed using C0042 instead of X0006.

Scan video and report (DDS) files for fields that are different than their data dictionary sizes.

What the Search Will Not Do

Data items that have been added to files must be added to the program and possibly to the display/report file.

Data structures defined in the program should be checked. If such a structure exists, and the file has changed, you may need to redefine the structure.

CASE-Generated Programs

The following topics review the programs provided to convert all existing CASE-generated programs. These programs are used for source that still have CAP status equal to 'Y'. The most significant change is the use of file servers for the data dictionary and the cost center master, as well as the use of a new file server for user defined codes.

Migrating CASE Generated Programs

Theoretically, when migrating CASE-generated programs from one release to the next, the procedures should be:

1. Change field sizes on the printer/display file.
2. Regenerate the source from the specifications.
3. Recompile all members.

For release A7, a conversion program is available to update the file specifications for new server technology. This conversion allows you to automate the inclusion of all supported server technology without removing specifications from each individual program.

CASE Conversion Process

To convert CASE-generated programs from A5 to A6 or A7:

1. Convert CASE specification files from A5 to A6 or A7.

   The CASE specification files (F93101, F93102, and so on.) are converted by the reinstallation process. If you need to transfer specifications manually, the Parameter Copy/Move program found on Menu A9361, Selection 14, allows the transfer of CASE specifications for a single program from an A5 library to an A6 or A7 library. You must use an A6 or A7 library list for this feature to work between environments.

2. Update the program types and logic modules.

   The bill of materials for program types and the master source for logic modules must be updated. The merge of these changes can be found on menu A9366, or as a selection from menu A97U1.

3. Convert the DDS for display files.

   All display files for CASE-generated programs must be converted to include the new DDS keywords. This conversion can be accomplished by using DREAM Writer version P89511. After converting the display files, they must be recompiled.

4. Convert CASE specifications.

   Because of the use of file servers for the data dictionary and the cost center master, references to F9800 and F0006 must be removed from the CASE specifications. The CASE conversion program removes all references to F9800, and references to F0006 when used for input only.

   The DREAM Writer version for the conversion of CASE specifications is P93995. The “Based on” File for DREAM Writer selection is F93102. The processing option allows for processing in proof or update mode.

5. Regenerate source and recompile.

   After completing the conversion, regenerate the source for each program and recompile the members. Use software version repository selections 15 (source generation) and 14 (compiling), or use the Global Program Regeneration selection found on menu A9361, and the Compile Multiple Objects selection found on menu A9362.

6. Convert the Source for CL Programs.

   There is new logic in the CL programs for language preference. This logic can be added by using the conversion programs for non-CASE software. The DREAM Writer version for conversion of CL programs is

P89501. The DREAM Writer selection should be the CL program name. After converting the source, recompile each member.

Accessing CASE Conversion Programs

You can find the CASE conversion program on menu A97U1. This conversion removes the specifications for files processed by servers from F93102 and F93103. It also removes the key field definitions from F93105.

You can also find CASE merge programs for logic modules and program types on menu A97U1. These merges update the F93000, F93001, and F93108 files for changes in master source.

Double-Byte Enablement

Double Byte Truncation Routine - C9822

All fields that can contain text material were redefined as “open” fields for Release A6 and forward by changing the field-type attribute to “O” in the data dictionary. Technically, a field defined as “open” allows for the entry of single-byte data, double-byte data, or a combination of both. The basic technical requirement is whenever a string of double-byte characters is entered into a text field, the double-byte character string must be bracketed with special characters. These special characters, called shift-out and shift-in characters, tell the system where the double-byte character string begins and ends. Without these delimiting characters, the system cannot resolve the correct meaning of the bits and bytes contained in a character string, and an AS/400 system error results. Program modifications must be made to process all field truncations of “open” fields through a special subroutine (C9822).

The C9822 subroutine checks the process of truncating a field containing double-byte data, making sure a double-byte character is not divided in half or cuts off one of the special shift-out or shift-in characters. If the subroutine detects such a condition, the truncated character string is adjusted to contain the last whole double-byte character, and a shift-in character is inserted.

How to Know When a Program Is Affected

Implementation of this subroutine is required.

All programs must be checked for any truncated open fields. If any such field truncations occur, those fields must be edited by using C9822.

Procedure

1. Insert copy module for E9822.

```
E************************************************************
E*
E*     Copy Composite Member for Common Subroutine - C9822
E*
E/COPY JDECPY,E9822
E************************************************************
```

2. Insert copy module for C9822.

```
C************************************************************
C*
C*   Copy Common Subroutine - Double Byte Truncation Routine.
C*
C/COPY JDECPY,C9822
C************************************************************
```

3. Modify field MOVEs for all fields being truncated as shown below:

```
H/TITLE PMODELDBCS - MODEL - Execute Text Truncation
H*-----------------------------------------------------------
C*
C*    Move Field xxx to Field zzz for Output. Execute C9822
C*
CSR                   CLEAR@UA
CSR                   MOVEA xxx      @UA
CSR                   Z-ADD yyy      #OUTLG
CSR                   EXSR C9822
C*                    ---- -----
CSR                   MOVEA@UB       zzz C*
C*-----------------------------------------------------------
```

Explanation of variables:

@UA	=	Input processing array
@UB	=	Output processing array
xxx	=	Field name that contains text to be truncated
yyy	=	Length of output field. (numeric value)
zzz	=	Display field

Performance Considerations for Programmers

What Makes an Application Run Slowly

The way programs use computer resources profoundly affects response time (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 hard to judge what will turn out to be “a lot of I/O” when the program runs at a customer site. Customers run our software against thousands or millions as many records as we have in our test data files. Customers set up their operation such that they incur many expensive instructions for each transaction record processed. They often run our software on computers much slower than our development machine.

We need to be aware of how we can make programs run faster and minimize the use of expensive resources. Because our software architecture is very modular (this is a popular concept in the software industry under the buzzword “object-oriented”), we can cause a lot of work to be done inadvertently that should be fairly simple processing.

Program Calls/Initialization

We use program calls extensively in our software. Some heavily-used programs are:

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

Some common subroutines perform program calls (for example C00161 and C0000). On the AS/400, calling a program is relatively expensive. If the program being called is a CL program, it is even more expensive. The best way to minimize this expense is to check whether we really need to call the program (again). For example, if the transaction date hasn’t changed, don’t convert it again. Just 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 will go 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 will still be 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 will still be 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
C0012	Right Justify Numeric Fields	This is a general-purpose subroutine, that 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 will 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 doesn’t need to be put through C0012.
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 will do the scrubbing. (It tests for ‘ ‘). This subroutine has been changed in A7.3 to test for ‘0’ or ‘ ‘.

Subroutine	Name	Explanation

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.
C00161	Format Numeric Fields for Output	This subroutine calls an Assembler program that will examine and edit your output (inserting decimal points). There are programs that execute this subroutine for alpha fields. It will not harm the alpha fields, but it is an unnecessary expense. We also recommend 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 show up as the most expensive thing our 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. We changed one batch program that did a lot of writes to use sequential-blocked writes. It reduced the run time by two thirds. See *Sequential I/O* for further discussion.

Don't 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.

Don't Read Records Unnecessarily

Don't "scan" the database. If we allow users to type record selection criteria in a subfile program, we should have supporting logical files so that we can read only the records they are interested in. We should 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 pick and choose the ones to look at (this is usually controlled by a processing option), don't 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 has been normalized (for example, the F0101 Address Book file was split into eight files in A7.1), don't assume that all the new file information is needed. Don't just replace CHAIN I0101 with CHAIN I0101, CHAIN I0112, CHAIN I0113, CHAIN 10114, CHAIN I0115, CHAIN I0116, CHAIN I0301, CHAIN I0401. Look at the fields that are 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, don't CHAIN every time to find it. An example is when you use next numbers to allocate a key for your file. Next numbers usually should come up with a key that does not exist in your file. If your file is defined as having a unique key, it is much less expensive to WRITE the new record with an error indicator on the WRITE (usually the write will succeed) than to CHAIN with the new key first (usually the CHAIN will fail).

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

For example, a control file has company as the major key. The customer has set up their control data for company 00000. The transaction records are for company 12345. For every transaction record, we 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. We don't find an eligible record, so we do the SETLL again with company 00000.

Because we encourage customers to do generic set up with company 00000, we could 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 through a file that has the record selection and sequencing done through DREAM Writer, you should be able to use sequential processing with that file. RPG allows the file to be opened for sequential-only processing if one of the following statements is true:

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

The only OPCODE against that file is a WRITE **

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

If you are reading through a file, and only every tenth or twentieth record is updated (or deleted), it would be best to open the file twice. (For example, open the physical for the READ, and open the logical for the update). Read through 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 Fnnnnnn  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 can't achieve true sequential processing for a file, you still can get some benefit from blocking the file with a different override:

```
OVRDBF Fnnnnnn NBRRCDS(xxxx)
```

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

For example, you read through a master file, and for each master record you do SETLL and READE in a transaction file. You will not get sequential-only processing on the transaction file, but you can still block it. If you read through 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 will run 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(nn) 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 will ignore this paging option if it determines that there aren’t enough memory or CPU cycles to use it.

Caching Control Files

Programs often access multiple control files for each transaction record processed. In order to cut down on reads to the control files, many programs have caching logic for them. Most of the file servers also have caching logic. 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 will do a LOKUP in the deduction-type array. If it doesn’t find the deduction type there, it will CHAIN to the deduction-type file. If it finds the deduction type in the file, it will add 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.

We have customers who stripped the caching logic out of a program and see significant improvement. We also have customers who added their own caching logic to a program that had none, and see significant improvement.

What Goes Wrong?

The biggest problem we face is that customers set up and use a variable number of control records. If we knew that every customer set up no more than 100 pay types, for example, the caching would work fine. Customers who set up more than 100 types may get no benefit from a cache with 100 entries. If we increase the size of the array to more than 100, it is no longer efficient. If we start trying to use "smart" logic to keep the 100 types most recently used in the cache, we end up spending more time managing the cache than we 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 they can get away with setting up a sales tax rate for company 00000 for the USA, they will. If not, they will set it up by state for company 00000. If that isn't specific enough, they will set it up by county for company 00000. Only as a last resort, they will set it up for each company separately. Because we don't know how granular their definition is, our 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), we will go through a cache search followed by all the failed CHAINs all over again for each transaction record. We need to store the answer in the cache the way the question was asked. We started out looking for the sales tax rate for company 12345 for Jefferson County. We found the answer as 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 we can populate the array correctly (for example, if we store 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 we have 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

We can tell the number of records in a control file by looking in the file information data structure when we open the file. (See copy book I00INFDS - field FIRCNT). If we are using a 100-entry array cache, and there are 100 records or less in the control file, we can load all entries in S999 into a sorted array. This will speed up every LOKUP to the array. The downside to this technique is that we will have to define two arrays: one sorted and one not sorted. If you define the array as a sorted array, but the contents are not in the correct sequence, the LOKUP will fail.

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

Expensive Instructions

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

Array Handling

Anything to do with array processing can be expensive because the machine executes instructions over and over again 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
               ENDIF
               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:

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

will perform poorly compared to:

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

The XFOOT OPCODE sums all the entries in an array. If you have sized your array for many more entries than it usually contains, any arithmetic operation that runs against the whole array will run 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 will do 250 ADDs, even though 200 of the entries in each array contain zero.

```
E             A              250  150
E             B              250  150
E             C              250  150
C         A      ADD                     B        C
```

It is important to size your arrays carefully.

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 will generate individual instructions to move blanks to the entire data structure, then move 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  40 ALPH1
I                                 P  41  430PKD1
I                                    44  73 ALPH2
I                                    74  810NUM2
```

```
C                     CLEAREXAMP
```

vs

```
C                     RESETEXAMP
```

NOTE: If you will be using RESET, and the data structure contains numeric fields, you must initialize the data structure with ‘I’ on the DS statement.

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

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

In handling the relevant entries for each of the arrays, the system has to calculate where the entry is in each array. If we 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  60 DRDL02
I                                       61  70 DRSPHD
 *
C           DRDS      LOKUPKEY,#A           66
C           *IN66     IFEQ '1'
C           #A        OCUR DSDESC
C                     MOVE DRDL01    SFDL01
C                     MOVE DRDL02    SFDL02
C                     MOVELDRSPHD    SFSPHD
```

Arithmetic

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

```
C           $FMTYR     DIV  4         $NBRV9  99
C           $NBRV9     IFEQ .000000000
C                      MOVE '1'       $LEAP
```
The division operation stored the result in a field with no integers defined. For example, 1985 divided by 4 equals 496.3, but this would be stored in the program as .3 because $NBRV9 was defined with no leading numbers. This is an overflow condition. The result field is too small to hold the calculated result.

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

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

```
CSR                   MOVE F@AD      #A
CSR                   DO   #A
CSR                   MULT 10        #@AD
CSR                   END
```
It would be more efficient to do the following:

```
CSR                   MOVE F@AD      #A
C                     SELEC
C           #A        WHEQ 1
CSR                   Z-ADD10        #@AD
C           #A        WHEQ 2
CSR                   Z-ADD100       #@AD
C           #A        WHEQ 3
                        :
                        :
C           #A        WHEQ 9
CSR                   Z-ADD1000000000#@AD
C                     ENDSL
```
Luckily, S998 is only executed once, but be aware of how you do your arithmetic in the code which is executed over and over again.

Error Handling

We need to have error-handling logic in our programs. We 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 go ahead and 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. Our 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 we see a print job being submitted every 5 minutes, we know the job initiation is using 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 (Videos)

Many of our customers have remote locations. Subtle changes in the way we define display files can have a big impact on response time.

Display files created before A5.2 all had the PUTOVR/OVRDTA keywords as a standard. This requires the use of the RSTDSP(*YES) option when creating the display file. The old-style windows that we used in A5.2 required the use of RSTDSP(*YES) as well. RSTDSP(*YES) can slow down response time for remote users. It also impacts 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, the 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. Our 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 get added to other users' response time, because they have to wait for it to complete before they get a shot 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 we need RSTDSP(*YES) is if we have PUTOVR/OVRDTA keywords in the display file DDS, or if we have an old-style window (does not use WINDOW keyword). Unfortunately, RSTDSP(*YES) is the default on the SVR record. If we don't type anything in the MAINT/RSTDSP field, it will compile the display file with RSTDSP(*YES).

Changing the RSTDSP attribute during testing is very 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 we specify RSTDSP(*YES) unnecessarily.

General Batch Considerations

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

Batch Window

The batch window is usually the off-shift time when most of the employees are home. The customer wants to get all batch jobs done before business opens for the day. This could be 6p.m. to 7a.m., or a shorter time. One of the ways we can help the customer get through the batch work is to design for multi-threading. We should 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, we 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, we need the data physically on the disk in the sequence in which we are reading it. SORT would be appropriate for a work file that is, for example, created by this batch job, and that does not have many logical files over it.

If you are creating and updating summary records in a database file as part of the batch run, consider processing the input file in the correct sequence to do "level break" logic. For example, you want to create a summary record by salesman in a territory. If you read the input records in that same sequence, you can accumulate totals for the salesman until the salesman changes. Then you would write out the summary record for that salesman, and so on. If you process the data in a different sequence, you will have to repetitively retrieve and update the summary record for the salesman. This is expensive. 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 through 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 will 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 would specify 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  62                        FY
FNU  63  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 will not select data for you.

Be wary of using MAINT(*REBLD) or MAINT(*DLY). With MAINT(*REBLD), the entire access path will be rebuilt every time. It can slow down your testing. With MAINT(*DLY), the system monitors the percentage of records changed and dynamically changes the file to MAINT(*REBLD) if you change more than 20%. Consider removing the logical file member instead of MAINT(*REBLD). It is more efficient.

```
RMVM FILE(F0101LA) MBR(F0101LA)
CALL  UPDPGM
ADDLFM FILE(F0101LA) MBR(F0101LA)
```

National Language Support

Beginning with A7.3, J.D. Edwards complies with IBM's V3R1 National Language Support features. A7.3 users now can take advantage of IBM's Character Data Representation Architecture (CDRA), which is not supported by prior J.D. Edwards releases. This chapter discusses the aspects of CDRA that are relevant to J.D. Edwards software, the J.D. Edwards approach to compliance, and guidelines for A7.3 programmers. For a thorough introduction to CDRA, refer to IBM's *AS/400 National Language Support* for V3R1.

CDRA Overview

CDRA is IBM's method of achieving "consistent representation, processing, and interchange of coded characters (data) in AS/400 business computing systems and across IBM systems" (*AS/400 National Language Support*, section 2.2). This is accomplished by tagging all character data with a coded character-set identifier (CCSID). Because character data is represented internally by hexadecimal code points, the data's CCSID tells the system how to interpret the hex values to arrive at its correct character representation. Therefore, if data coded to one CCSID is read by a job coded with a different CCSID, the integrity of the character data is preserved as CDRA handles the conversions of hex code points under the covers.

The *invariant character set* (see *Invariant Character Set* in this chapter) is a special set of characters that are mapped to the same code points in virtually all code pages. Therefore, their hex values never change even if they are transmitted across different CCSIDs. We refer to characters outside of this set as *chameleons* because their hex code points will change to adapt to different CCSIDs in order to retain the same graphic character representation. Chameleons common to the US code page are '$', '#', '@', '!', and '¢'.

J.D. Edwards Implementation

The J.D. Edwards approach to tagging file data is to distinguish between textual data and non-textual data at the field level. Those fields that contain text (marked with an open data type in the data dictionary) are tagged with CCSID 00037, which is the CCSID of our development AS/400. This CCSID is generally used by North American AS/400s. Those character fields that are not used to store descriptive text (marked with an alpha data type in the data dictionary) are tagged with CCSID 65535. This CCSID indicates that the data is to be treated as hexadecimal data rather than coded graphic character data. In this way, we preserve the graphic character integrity of our textual data while also preserving the hexadecimal integrity of our non-textual character data.

Initially, we are inserting CCSID keywords in the DDS for all non-numeric fields in physical files. The File Design Aid has been enhanced to automatically insert these keywords when exiting so that the programmer does not need to be concerned about it. When IBM incorporates the CCSID parameter into the field reference file, we will delete the keywords from the file DDS and insert them into the field reference files instead.

In addition, we changed the create option (14) in the software versions repository to submit the creation with a job CCSID of 00037. This will cause all source data for non-ILE programs, and constant data for display and printer files, to be interpreted through a 00037 lens. ILE program source data will be interpreted according to the CCSID of the source file. Our source files are tagged as 00037. Incidentally, the IBM compilers for non-ILE programs interpret data in source files through a 00037 lens. "Most compilers expect syntactical operators and the naming convention for the source code to be in code page 00037; therefore, undesired mapping will occur if the source is compiled with a CCSID other than 00037 or 65535"(*AS/400 National Language Support Planning Guide* section 3.4.10).

We also changed the display and printer file create commands to include the *JOBCCSID value for the CHRID parameter (default is *DEVD). This will preserve character data integrity for display and printer files where the user's job CCSID is different from the user's device description CHRID.

Programming Guidelines

Follow these guidelines when you modify J.D. Edwards objects, or when you create objects by using the J.D. Edwards FDA, RDA, SDA, or CASE tool.

Do not change any file CCSIDs either at the file or the field level.

If you create a source file to use for J.D. Edwards programs or with J.D. Edwards programming utilities, be sure to tag it with a CCSID of 00037. A source file CCSID defaults to the CCSID of the job that created it.

If you modify a source member through PDM:

For physical file DDS, do not change the CCSID keyword values. Otherwise, you can easily destroy data integrity. If you want to add a field, use File Design Aid, which will automatically insert the appropriate CCSID attribute for you.

When you create an object through PDM, be sure your job CCSID is 00037. A user's job CCSID defaults to the user profile CCSID. To change it, enter CHGJOB CCSID(00037).

When you create a display file or printer file through PDM, be sure the CHRID keyword has value *JOBCCSID. This is *not* the default value.

When you create interactive programs, never require the user to enter a chameleon character in order to perform a particular function. Because the program looks for a specific hex value, the graphic character to enter will vary according to the user's job CCSID. Therefore, this feature cannot be documented. Hints as to the location of any existing cases can be found in vocabulary overrides and processing options. (Example to avoid: "Enter '@' to view all types".)

Do not embed data dictionary item names in text (for example, glossary and processing option text) if they contain chameleon characters. (Example to avoid: "Default value from #CYR in data dictionary".)

Do not assign chameleon characters to data dictionary default or allowed values, or to processing options default values.

Do not use chameleons as or within literals in source code. (Example to avoid: FLDA IFEQ '$AB'.)

Invariant Character Set

The following characters are included in the invariant character set:

26 unaccented uppercase letters 'A' through 'Z'

26 unaccented lowercase letters 'a' through 'z'

Ten digits '0' through '9'

Plus sign

Less-than sign

Equal sign

Greater-than sign

Percent sign

Ampersand

Asterisk

Straight double quote

Straight single quote

Left parenthesis

Right parenthesis

Comma

Underscore

Hyphen

Period

Slash right

Colon

Semicolon

Question mark

A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	a
b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s
t	u	v	w	x	y	z	0	1
2	3	4	5	6	7	8	9	+
t	+	u	%	&	*			(
)	,	_	-	.	/	:	;	?