JD Edwards EnterpriseOne Tools

Auditing Administration Including 21 CFR Part 11 Administration Guide

Release 8.98 Update 4

E14690-02

March 2011

ORACLE®

JD Edwards EnterpriseOne Tools Auditing Administration Including 21 CFR Part 11 Administration Guide, Release 8.98 Update 4

E14690-02

Contents

Preface

Welcome to the JD Edwards EnterpriseOne Tools Auditing Administration Including 21 CFR Part 11 Administration Guide.

Audience

This guide is intended for system administrators and technical consultants who are responsible for setting up auditing and electronic signatures.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Computer desktop application usage and terminology.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/us/corporate/accessibility/index.html`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

You can access related documents from the JD Edwards EnterpriseOne Release Documentation Overview pages on My Oracle Support. Access the main documentation overview page by searching for the document ID, which is 876932.1, or by using this link:

`https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=876932.1`

To navigate to this page from the My Oracle Support home page, click the Knowledge tab, and then click the Tools and Training menu, JD Edwards EnterpriseOne, Welcome Center, Release Information Overview.

This guide contains references to server configuration settings that JD Edwards EnterpriseOne stores in configuration files (such as jde.ini, jas.ini, jdbj.ini, jdelog.properties, and so on). Beginning with the JD Edwards EnterpriseOne Tools

Release 8.97, it is highly recommended that you only access and manage these settings for the supported server types using the Server Manager program. See the Server Manager Guide on My Oracle Support.

Conventions

The following text conventions are used in this document:

Convention	Meaning
Bold	Indicates field values.
Italics	Indicates emphasis and JD Edwards EnterpriseOne or other book-length publication titles.
`Monospace`	Indicates a JD Edwards EnterpriseOne program, other code example, or URL.

1 Introduction to Auditing Administration for JD Edwards EnterpriseOne

This guide describes the tools necessary to administer auditing and electronic signatures for JD Edwards EnterpriseOne.

Oracle's JD Edwards EnterpriseOne auditing and electronic signature tools provide a solution to the Food and Drug Administration's (FDA) acceptance of electronic signatures and audit records for FDA-required records such as product submissions, batch records, and complaints. These tools enable your organization to comply with the FDA 21 CFR Part 11 regulation for submitting electronic records.

2 Configuring JD Edwards EnterpriseOne for Auditing and Electronic Signatures

This chapter contains the following topics:

2.1 Understanding Audit and Electronic Signatures Records

JD Edwards EnterpriseOne provides the capability to select specific columns in a table for auditing. You can configure JD Edwards EnterpriseOne to generate an audit table when table records are inserted, updated, or deleted. The audit records contain data such as:

- Before and after values.
- Time and date of the transaction.
- The user who made the modification.

In addition, you can configure JD Edwards EnterpriseOne interactive and batch applications to require an electronic signature approval when a user tries to change the data on an application or submit a report. A record of the approval is recorded in the Signature table (F9500005). The table records this information:

- Approver of the change.
- Reason for the approval.
- Approver's user ID.
- User's role.
- Date and time of the approval.

You can view the information in the audit and electronic signature tables using the View Audit/Signature Information application (P9500005) or by generating reports (R9500004, R9500005, R9600006). The reports display all the audit and signature information in an easy-to-read Adobe Acrobat PDF file that can be printed to hard copy or saved in digital format.

2.2 Prerequisites

Before you configure JD Edwards EnterpriseOne for auditing and electronic signatures:

- Use Security Workbench (P00950) to secure the audit applications (P9500001, P9500002, P9500003, and P9500005) in JD Edwards EnterpriseOne to ensure that only the appropriate users have access.
- Create a backup of the database before configuring audit tables and make sure that the restore process works.
- Plan the audit:
  - Research and determine which tables and columns need to be audited.
  - Research and determine if database triggers and database views are currently in use for the tables that you plan to audit. If this is the case, you must manually remove those objects before auditing is configured. Once auditing is configured, those objects must be manually enabled.
  - Research and determine which applications, forms, and reports require electronic signature approvals.
  - Research and determine which roles are required for electronic approval. If needed, set up additional roles in JD Edwards EnterpriseOne.

    See "Setting Up Roles" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.
  - Plan for additional disk space for audit tables.
- Make sure that the proposed audited table names are not called tablename_ADT.

  If they are, rename them.

2.3 Setting Up JD Edwards EnterpriseOne for Auditing and Electronic Signatures

You must perform the tasks in this section before you can configure JD Edwards EnterpriseOne tables and applications for auditing and electronic signatures.

This section provides overviews of database considerations for auditing and database triggers for audited tables, and discusses how to:

- Set up a data source for audit tables.
- Configure an Oracle database for auditing.
- Configure a DB2 for IBM i database for auditing.
- Configure a IBM DB2 for LUW (Linux, UNIX, Windows) database for auditing.

2.3.1 Understanding Database Considerations for Auditing

This section discusses:

- Database sizing.
- Table conversions.
- Database driver considerations.
- Relational database management system (RDBMS) interface management.

2.3.1.1 Database Sizing

When auditing is enabled in JD Edwards EnterpriseOne, the size of the database increases dramatically. In addition to creating six new tables, an audit table is created for every table that has a column flagged for audit. You can use the following formula to calculate the increase in database size:

Audit table record size = 102 characters + size of allaudited columns, including primary key for an audited table

For every record inserted or deleted on a table that is being audited, the system inserts a record in an audit table. For each update transaction on a table that is being audited, the system inserts two records in the audit table: one for the before image and one for the after image. Use the following formula for update transactions:

Database size increase per audited table = audit tablerecord size × 2 (before and after image) × number of transactions

2.3.1.2 Table Conversions

If you perform a table conversion when auditing is turned on, the tool audits the conversion itself, slowing the table conversion process. To prevent this from happening, disable auditing before performing a table conversion.

2.3.1.3 Database Driver Considerations

The database driver level implements triggers on the database tables to provide auditing of selected tables, for both changes made through JD Edwards EnterpriseOne database middleware and changes made by third-party software. The auditing includes the recording of signature information when this feature is activated. The auditing information is passed to the drivers from JDB using additional table columns. Database level table triggers generate the auditing information for third-party changes.

The database triggers record auditing information for every INSERT and DELETE statement. For UPDATE statements, the triggers record auditing information only if one or more of the audited columns have a changed data value. Modifying a non-audited column does not generate any audit information.

For a SQL server, a column cannot be dropped if it is:

- A replicated column.
- Used in an index.
- Used in CHECK, FOREIGN KEY, UNIQUE, or PRIMARY KEY constraints.
- Associated with a default defined with the DEFAULT keyword or bound to a default object.
- Bound to a rule.

2.3.1.4 RDBMS Interface Management

This table lists the RDBMS interface associated with each database:

Database	**RDBMS Interface**
Oracle	Enterprise Manager
SQL Server	Query Analyzer
DB2 for IBM i	Primitive Support

Database	RDBMS Interface
IBM DB2 for LUW (Linux, UNIX, Windows)	Control Center

2.3.2 Understanding Database Triggers for Audited Tables

The database triggers for audited tables write records to the audit tables (shadow tables) for Insert and Delete functions. For Insert, the after image is stored in the audit table. For Delete, the before image is stored in the audit table.

For the Update function, the database triggers only write records to audit tables when either the primary key columns or the specified audited columns are changed. A record is written in the shadow table when the new value is different from the current value. No record is written if they are the same. If the trigger is executed for update, before and after images of the record change are recorded in the audit table.

The trigger templates for all databases are stored in the Database Object Template Information table (F986112).

2.3.2.1 Oracle

Each audited table has five triggers, three of which are ROW level triggers (Before for Insert and Update, After for Delete) and two of which are STATEMENT level triggers (Before/After). Each signature table has one trigger and each audit table has one package.

2.3.2.2 DB2 for IBM i

Each audited table has three ROW level (After) triggers. Two triggers are created for each signature table.

2.3.2.3 IBM DB2 for LUW (Linux, UNIX, Windows)

Each audited table has three ROW level (After) triggers. Two triggers are created for each signature table.

2.3.2.4 SQL Server

Each audited table has three triggers, which are the STATEMENT level triggers (After). One trigger is created for each signature table.

2.3.2.5 Trigger Naming Conventions

This table contains the trigger naming convention for each database type:

Database	Object Name	Object Attribute
Oracle	Fxxxx_CFRA_RDA	Row Level Delete Trigger
Fxxxx_CFRA_RIA	Row Level Insert Trigger	N/A
Fxxxx_CFRA_RUA	Row Level Update Trigger	N/A
Fxxxx_CFRA_SUA	Statement Level Update Trigger	N/A
Fxxxx_CFRA_SUB	Statement Level Update Trigger	N/A
Fxxxx_ADT_CFRA_PKG	Package	N/A

Database	Object Name	Object Attribute
F95xxxxx_CFRS_RIUB	Signature/Audit Header/Audit Detail Table Insert and Update Triggers	N/A
DB2 for IBM i and IBM DB2 for LUW (Linux, UNIX, Windows)	Fxxxx_CFRA_RDA	Row Level Delete Trigger
Fxxxx_CFRA_RIA	Row Level Insert Trigger	N/A
Fxxxx_CFRA_RUA	Row Level Update Trigger	N/A
F95xxxxx_CFRS_RIB	Signature / Audit Header / Audit Detail Table Insert Triggers	N/A
F95xxxxx_CFRS_RUB	Signature / Audit Header / Audit Detail Table Update Triggers	N/A
SQL Server	Fxxxx_CFRA_RDB	Statement Level Delete Trigger
Fxxxx_CFRA_RIB	Statement Level Insert Trigger	N/A
Fxxxx_CFRA_RUB	Statement Level Update Trigger	N/A
F95xxxxx_CFRS_RIB	Signature/Audit Header/Audit Detail Table Insert and Update Triggers	N/A

2.3.3 Setting Up a Data Source for Audit Tables

Make sure that you install 21 CFR Part 11 objects based on the JD Edwards EnterpriseOne release level.

1. Sign on to a client machine.
2. Create a non-Julian database data source for system tables.

   For example, create "System - B9NJ" data source for non-Julian tables for JD Edwards EnterpriseOne 8.11.
3. Generate tables using Object Management Workbench (OMW) for the respective data sources (except F9500005).

   The System Data Source name is based on the JD Edwards EnterpriseOne release level; for example, "SYSTEM - B11" for JD Edwards EnterpriseOne 8.11 and "SYSTEM - 812" for JD Edwards EnterpriseOne 8.12.

Tables	Data Sources
F9500001	System Data Source
F9500002	System Data Source
F9500003	Non-Julian System Data Source
F9500004	Non-Julian System Data Source
F9500005	Local - XXXX
F9500006	Non-Julian System Data Source
F986112	System Data Source

4. Create Object Configuration Manager (OCM) mappings for *PUBLIC in the System and Server Map using this information:

Tables	Data Sources
F9500001	System Data Source
F9500002	System Data Source
F9500003	Non-Julian System Data Source
F9500004	Non-Julian System Data Source
F9500005	System Data Source
F9500006	Non-Julian System Data Source
F986112	System Data Source

5. Map these objects to the appropriate data source:

Objects	Data Sources
B9500001	LOCAL
B9500003	LOCAL
B9500005	SERVER
B986112	LOCAL
R9500004	LOCAL
GT9500006	Object Librarian

6. Open Microsoft Windows Explorer and double-click the dbtemplates.exe, which is located in this directory:

```
C:\B9\system\bin32\
```

2.3.4 Configuring an Oracle Database for Auditing

You must grant the owners of the audit tables (F9500003, F9500004, and F9500006) and audited tables (for example PRODDTA, DD812) the privileges listed in this task.

1. Start SQL*PLUS and connect to the instance as SYS.
2. For each owner, enter the following three commands:
   – GRANT CREATE PROCEDURE TO owner
   – GRANT CREATE TRIGGER TO owner
   – GRANT SELECT ON SYS.V_$SESSION TO owner

Note: The third command must be directly granted to the owner. Granting it to a role to which the owner belongs does *not* work.

2.3.5 Configuring a DB2 for IBM i Database for Auditing

In an auditing configuration, when auditing is disabled for a table, all working columns will be dropped in the table through the SQL operation ALTER TABLE DROP COLUMN. In DB2 for IBM i, an Inquiry Message will be issued for this SQL operation when the column contains data. The purpose of the Inquiry Message is to obtain the user's response to carry on the operation or to cancel the operation. However, the system cannot send this Inquiry Message to the user for a response and this causes the system to automatically cancel the SQL operation. The consequence is failure to disable auditing for the table.

To resolve this issue, perform these setup steps:

1. Add the following entry in the system reply list for the Inquiry Message - CPA32B2:

   ```
   ADDRPYLE SEQNBR(3283) MSGID(CPA32B2) RPY('I')
   ```

2. If you are using iSeries version V5R2 or prior, modify the job description QDFTJOBD that is the *JOBD for iSeries User QUSER by changing the value for its Inquiry Message Reply option:

   ```
   CHGJOBD JOBD(QGPL/QDFTJOBD) INQMSGRPY(*SYSRPYL)
   ```

3. If you are using iSeries version V5R3 or later, modify the job description QDFTSVR that is the *JOBD for iSeries User QUSER by changing the value for its Inquiry Message Reply option:

   ```
   CHGJOBD JOBD(QGPL/QDFTSVR) INQMSGRPY(*SYSRPYL)
   ```

Note: Whenever you change the configuration of the iSeries, you must delete the SQL packages.

2.3.5.1 Deleting SQL Packages

To delete the SQL packages:

1. Stop services with this command:

   ```
   ENDNET
   ```

2. Delete the SQL packages with this command:

   ```
   WRKOBJ *ALL/*ALL *SQLPKG
   ```

   Delete all *SQLPKG objects whose name does not start with "Q".

3. Start the service with this command:

   ```
   STRNET
   ```

2.3.6 Configuring a IBM DB2 for LUW (Linux, UNIX, Windows) Database for Auditing

To configure a IBM DB2 for LUW (Linux, UNIX, Windows) database for auditing, you must set up the privileges described in this section.

1. Assign at least one of the following privileges to the owners of the audited tables:
   - SYSADM or DBADM
   - ALTER privilege on the table on which the trigger is defined, or ALTERIN privilege on the schema of the table and one of the following privileges:

     -CREATEIN privilege on the schema.

     -IMPLICIT_SCHEMA authority on the database.

2. Assign the following privileges to all of the JD Edwards EnterpriseOne database proxy users:
   – SELECT privilege on the table.
   – SELECT privilege on any other table or view that the trigger is referencing.
   – Necessary privileges to execute the trigger SQL statement.

2.4 Preparing an Auditing and Electronic Signature Configuration for a JD Edwards EnterpriseOne Upgrade

Before upgrading to a new version of JD Edwards EnterpriseOne or applying software updates that contain table changes, you must turn off the auditing and electronic signature functions. JD Edwards EnterpriseOne must be made unavailable to users, and all the audit tables and configuration tables (F9500001 through F9500006) must be archived.

Note: Audit history will *not* be migrated when JD Edwards EnterpriseOne is upgraded. You must reconfigure the table audit and electronic signature settings.

1. Make sure that users are signed off from the JD Edwards EnterpriseOne installation that is configured for auditing and electronic approval.
2. Create a backup of the tables that are being audited, shadow audit tables, and auditing configuration tables in a different data source.
3. Manually remove any custom database views or triggers.

   If a custom table trigger already exists, it will be deleted when auditing is enabled or disabled.
4. Disable auditing using the Configuration Application (P9500001).

   This application performs these functions:
   – Drops native database triggers on Fxxxx_ADT.
   – Drops database view Fxxxx.
   – Drops CFR columns for Fxxxx_ADT.
   – Renames Fxxxx_ADT to Fxxxx.
5. Perform the JD Edwards EnterpriseOne upgrade.
6. Enable auditing using the P9500001 application.

   This application performs these functions:
   – Renames target table to Fxxxx_ADT.
   – Adds CFR columns to Fxxxx_ADT.
   – Creates database view Fxxxx (original structure)
   – Creates native database triggers to Fxxxx_ADT.

Note: Remember that before you enable auditing, you must perform certain tasks on the JD Edwards EnterpriseOne servers and database servers, as previously discussed in this chapter.

7. Recreate and manually enable the custom database view or triggers removed in step 3.

3 Setting Up Auditing and Electronic Signatures

This chapter contains the following topics:

3.1 Understanding Auditing and Electronic Signature Setup

This section discusses:

- Auditing for interoperability transactions
- Audit table setup.
- Table Design Aid in audit mode.
- Table date and time data.
- Table naming conventions.
- Tables associated with auditing and electronic signatures.
- Electronic signature setup.
- Electronic signatures for power forms and subforms.
- Audit records, signature records, and globally unique identifiers (GUID).

3.1.1 Auditing for Interoperability Transactions

An interoperability transaction can affect a column in a JD Edwards EnterpriseOne table that has been enabled for auditing. When this occurs, JD Edwards EnterpriseOne creates an audit record for the transaction, but the system only records a portion of the audit information, such as the audited column, before and after values, and recorded columns. The audit information will not include a GUID, application ID, workstation name, or IP address, unless you configure the interoperability model to pass this data to the audit record.

See Configuring Auditing for Interoperability Transactions.

3.1.2 Audit Table Setup

You use the Configuration Application (P9500001) to select the table that you want to audit. After selecting a table, JD Edwards EnterpriseOne launches Table Design Aid (TDA). In TDA, you select the actual columns in the table that you want to audit.

When the auditing function in JD Edwards EnterpriseOne is turned on, the system records changes, additions, and deletions to the data in these columns in an audit table (shadow table) to provide a tracking history. An audit table contains the columns used as the primary key of the original table along with the columns that you define to trigger an audit. In addition, you can select columns for recording. Recorded columns do not trigger an audit, but appear in an audit table along with the other columns that trigger an audit.

Every time a change is made to an audited column in the original table (Fxxxx), the audit table (Axxxx) records this information:

- The field that was changed.
- The previous value in the field.
- The new value in the field.
- The date and time of the change.
- Any other fields in the table that were selected for auditing.
- Any other fields in the table that were selected for recording.
- The form, application, and version name used to make the change.
- The user ID and internet protocol (IP) address of the person who made the change.

The auditing function can be turned on or off for each specific table. Even the action of turning the auditing function on and off is recorded in a table to provide a history of the audit tables.

3.1.3 Table Design Aid in Audit Mode

When TDA is launched in audit mode, it displays the default columns of the table that you are auditing in read-only mode. You can drag and drop columns that you want to audit from the audited table (Fxxxx) to the audit table (Axxxx). BLOB (Binary Large Object) columns are not displayed in audit mode and cannot be audited.

All primary key columns of the audited table are automatically added to the audit table. The default audit table columns are automatically added to the audit table as well. If a default index does not exist, TDA creates one for the audit table and populates it with CFRGUID and B4ORAFTR from the default audit table columns, along with the primary key index columns.

3.1.4 Tables Associated with Auditing and Electronic Signatures

This section describes each of these tables associated with auditing and electronic signatures:

- Configuration Table (F9500001)
- Reason Codes (F9500002)
- Audit Header (F9500003)
- Audit Detail (F9500004)
- Temporary Audit (F9500005)

- Signature (F9500005)
- Audit (Axxxx)
- Database Object Template Information (F986112)

3.1.4.1 Configuration Table (F9500001)

The F9500001 table stores the configuration information that is entered using the Work with Configuration Settings form in the P9500001 application. The F9500001 table contains the object name, form or version, the event used, environment, the audit table, who to report realtime information to, number of signatures required, and the reason codes associated with the audit.

3.1.4.2 Reason Code Table (F9500002)

The F9500002 table stores the reason code information that is entered using the Work with Reason Codes form in the Reason Codes application (P9500002).

3.1.4.3 Audit Header Table (F9500003)

The F9500003 table stores the audit header information. This information includes the unique ID, user ID, user's full name, user's address book number, the workstation from where the change was made, date, time, and program ID.

3.1.4.4 Audit Detail Table (F9500004)

The F9500004 table stores the audit detail information. This information includes the unique ID, table name, audit table name, the action used, user ID, workstation from where the change was made, date, time, and program ID.

3.1.4.5 Temporary Audit Table (F9500005)

The F9500005 table temporarily stores the audit information. Results from queries or Universal Batch Engines (UBEs) run on the client are stored in this table and used for reporting.

3.1.4.6 Signature Table (F9500006)

The F9500006 table stores the captured signature information. One F9500006 table exists per instance of JD Edwards EnterpriseOne. This table records a unique ID, the object, the form or version name, the user ID, the user's full name, their address book number, the Reason Code they choose, and their group name.

Electronic signature failures are written to the F9500006 table only after four failed authorization attempts. The authorization is recorded as a failure, and a notification is sent to the person identified in the Real Time Notification configuration, if enabled.

3.1.4.7 Audit Table

This table stores the audit information. When auditing is turned on for a particular table, the system creates an audit table that contains the primary keys of the audited table, audited columns, recorded columns, and these default columns:

Data Item Name	Alias	System	Type	Size	Populated By	Value
CFR GUID	CFRGUID	H95	String	36	Trigger	■ Actual GUID if auditing is on with OK or Delete event. ■ "OW Without GUID" (events other than OK or Delete) ■ Hard-coded with "Third Party" for third party.
Primary Key Columns of Audited Table	N/A	N/A	N/A	N/A	JD Edwards EnterpriseOne or third-party	Primary key value
Before or After	B4ORAFTR	H95	Character	1	Trigger	■ A for after image. ■ B for before image.
Action	CFRACTION	H95	Character	1	Trigger	■ 1 for insert ■ 2 for update ■ 3 for delete
Audited Columns	N/A	N/A	N/A	N/A	JD Edwards EnterpriseOne or 3rd party	Data being audited
User ID	CFRUSER	H95	String	10	JD Edwards EnterpriseOne or Trigger	UserID
Machine Key	CRFMKEY	H95	String	15	JD Edwards EnterpriseOne or Trigger	MachineID
Date/Time	DATETIME	H95	String	26	Trigger	Actual date/time
Program ID	CFRID	H95	String	10	JD Edwards EnterpriseOne or Trigger	Program ID
Audit Type	AUDITTYPE	H95	Character	1	Trigger	■ 1 for JD Edwards EnterpriseOne with GUID ■ 2 for "OW without GUID" ■ 3 for third-party

The audit table must be in the same data source as the parent table.

Note: If an application or business function (BSFN) runs in a transaction boundary, and the transaction gets rolled back, the audit table may not contain any changes that are rolled back. Because the Audit Header and Audit Detail tables are not included in the scope of the transaction, records written to these two tables are not rolled back.

3.1.4.8 Database Object Template Information table (F986112)

The F986112 table stores trigger templates. This table is populated by the DBTemplates.exe program. The templates in F986112 are used by the database drivers to create native database triggers.

3.1.5 Table Date and Time Data

The Audit Header (F9500003), Audit Detail (F9500004), and Signature (F9500006) tables have the date and time in two separate columns. The data is populated using native database triggers. The date consists of the year, month, and day. The time consists of the hour, minute, and second. These tables must be mapped to the same data source to ensure that they have the date and time from a central source. This data source must be a non-Julian data source.

For all audit tables, date and time are combined into a single string column. The native database trigger populates year, month, day, hour, minute, and second.

3.1.6 Table Naming Conventions

When audit table triggers are created, JDB alters the audited table by adding audit-related columns and any columns that native database triggers need in order to perform an audit trail. However, this addition changes the physical metadata for the audited table, making it different from the table's JD Edwards EnterpriseOne specification.

For JD Edwards EnterpriseOne applications and other third-party applications to function properly, the audited table is renamed. A new database view is created for that table using the original audited table name. For example, when auditing is enabled for the Address Book Master table (F0101), the audit table triggers are created on the F0101 table. JDB renames F0101 to F0101_ADT. A database view named F0101 is created for the F0101 using all the existing JD Edwards EnterpriseOne columns. The new renamed table, F0101_ADT, is altered to add all audit-related columns, and a set of native database triggers are created on F0101_ADT to send any changed records to the audit table (A0101 table). This approach enables JDB to pass audit information to the native database triggers. At the same time, access to the audited table, either through third-party software or a JD Edwards EnterpriseOne application that queries metadata, can function properly without any modification to the application.

Note: Because audited tables add _ADT to the table name, any existing tables ending with _ADT are assumed to be audited tables. Before installing the software, check all databases for tables with _ ADT and rename them.

When disabling auditing for a particular table, the operation is the reverse of enabling auditing. JDB puts the table back to the stage where it was before the auditing was enabled.

3.1.7 Electronic Signature Setup

You can configure interactive and batch applications to require an electronic signature approval to complete a transaction or launch a report. You can configure electronic signatures for an application only if auditing has been enabled for the same application.

The electronic signature consists of the user ID and password of the approver. When an approver enters a valid electronic signature, JD Edwards EnterpriseOne records the approver's information in the Signature table (F9500006).

JD Edwards EnterpriseOne verifies an electronic signature by checking that the user is a JD Edwards EnterpriseOne user, that the role is acceptable, and that the user ID and password are valid. If the electronic signature fails, an error is generated. After four failed attempts, the user is no longer given the opportunity to enter an electronic signature, and the attempt is logged in the F9500006 table. No progress is allowed without a valid signature.

You can use up to two roles to define those users allowed to enter electronic signature approvals for a given application. For instance, you may have a business process that calls for two electronic signatures to approve a transaction. You can assign two roles to the signature configuration: a role for the first approval and a role for the second approval. The roles can include users who have different levels of authority in the organization, such as a role that includes supervisors and another role that includes managers.

You can turn on or turn off the electronic signature function at the pathcode level. Whenever a change is made in the state of the signature function, it is recorded in the Configuration Table (F9500001).

3.1.7.1 Reason Codes

All electronic signature approvals require a reason code. Reason codes are used to denote the reason for an electronic signature approval. When an approver enters an electronic signature, they must also select a reason code for the approval. Using the Reason Codes application (P9500002), you can create new reason codes, view current reason codes, activate or deactivate reason codes, and define default reason codes. You can define a default reason code for an application, although all reason codes are available for each application and can be easily changed.

Existing reason codes cannot be edited or deleted, but their status can be changed to either active or inactive. This action ensures that obsolete reason code descriptions are still available for older audit records. In addition, you cannot change the status of a default reason code.

3.1.7.2 Electronic Signature Features

You can use any of the following options when configuring electronic signatures for an application:

Option	Description
Real Time Notification	This option informs a user or administrator when an approval fails due to an incorrect password, user ID, or permission. Four approval failures for the required signature will generate a notification message. When you configure the application for an electronic signature approval, you can specify the address book number or email address of the person to be notified. You can configure a different user to receive a notification for each application that you configure to require a signature.
Pre-populate Approver	This option automatically fills in the user's JD Edwards EnterpriseOne sign-in information for the first and second approver, depending on the configuration for the number of approvers.
Pass Through User	This option records the user's JD Edwards EnterpriseOne sign-in information when using a specified application. This function is only used to record the person using the application, the time, and from what machine. It does not prompt the user for an electronic signature approval. It is only a tracking device and cannot be used with the Real Time Notification or Pre-populate Approver functions.

3.1.7.3 Considerations for Batch Applications

Configuring electronic signatures for batch applications is the same as configuring them for interactive applications. However, before you configure an electronic signature for a batch application, you should consider how the batch application is launched. Batch applications can be launched by users from JD Edwards EnterpriseOne menus, or they can be automatically launched by event rules attached to a button in an interactive application. If an OK, Save, or Delete button is configured for an electronic signature approval and that same button contains an event rule that launches a batch application, the system will not launch the batch application until a signature approval is entered for the interactive application. In this case, the system generates a signature record for the transaction on the interactive application, but not the batch application. If a business process requires a signature record each time a batch application is run, then you must create an electronic signature configuration specifically for the batch application.

Consider these different scenarios when configuring electronic signatures for batch applications that are launched by event rules attached to an OK, Save, or Delete button:

- If the button and batch application are configured for different signatures, then the system will prompt for two separate signature approvals.

  The system records two signature records, one for the interactive application and one for the batch application.

- If the button and batch application are configured for the same signature, then the system prompts for a signature for the interactive application and then automatically uses that signature for the signature approval of the batch application. In this case, the system still records two signature records, one for the interactive application and one for the batch application.

- If the button is *not* configured for an electronic signature but the batch application is, then the system only prompts for an electronic signature for the batch application. The system creates only one electronic signature record.

3.1.8 Electronic Signatures for Power Forms and Subforms

JD Edwards EnterpriseOne enables you to configure the following types of power forms and subforms for electronic signatures:

- Power browse forms
- Power edit forms
- Reusable browse subforms
- Reusable edit subforms

Note: For auditing, there is no special configuration for power forms and subforms because auditing is configured at the table level. A column that is configured for auditing will trigger an audit regardless of the type of form it resides in.

You must have a thorough understanding of how power forms and subforms operate before you configure them for electronic signatures. Power forms can contain multiple subforms, embedded or reusable. An OK, Delete, or Save button on a power form can programmatically invoke a Delete or Save button on a subform, and vice versa. You can configure the system to require an electronic signature on a power form, a reusable subform, or both.

3.1.8.1 Embedded Subforms

You cannot explicitly configure an embedded subform for an electronic signature. Embedded subforms take on the electronic signature configuration of their hosting power form. Therefore, the embedded subform's signature configuration is considered implicit. If a user clicks the Save button on an embedded subform within a power form that has been configured for an electronic signature, the system will prompt for an electronic signature.

When you search on power forms and subforms to configure for electronic signatures, JD Edwards EnterpriseOne only displays power forms and reusable subforms, not embedded subforms.

3.1.8.2 Reusable Subforms

You can explicitly configure reusable subforms for electronic signatures. A reusable subform configured for an electronic signature acts independently of its hosting power form's configuration. You can configure the reusable subform to prompt for an electronic signature without configuring the power form.

When a reusable subform is not configured for an electronic signature and its hosting power form is, the reusable subform is considered to have an implicit configuration. In this case, the reusable subform operates in the same manner as an embedded subform.

In addition, you can explicitly turn off, or exclude, electronic signatures on a reusable subform. In this case, if a user clicks the Save button on the reusable subform, the system does not prompt for an electronic signature, even if the power form was configured for an electronic signature. For example, you may have a business process that requires you to configure a power form for an electronic signature. However, the power form contains a reusable subform that does not require an electronic signature. In this case, you would explicitly configure the reusable subform to exclude the electronic signature. As a result, when a user clicks the Save button on the reusable subform, the system will not prompt for an electronic signature for the subform.

Note: You cannot configure electronic signatures at the application level if the application contains a reusable subform.

3.1.8.3 Examples of Electronic Signature Configurations on Power Forms and Subforms

The following table describes the different ways that you can configure power forms and subforms for electronic signatures:

Note: The scenarios use the OK and Save buttons as examples, but you can also apply these configurations to the Delete button.

Configuration	Result
On a power form, the OK button is configured for an electronic signature. None of the subforms within the power form are configured for electronic signatures. The electronic signature configuration for these subforms is considered implicit.	Scenario 1: When a user makes changes on the power form and clicks the OK button, the system prompts for an electronic signature. Scenario 2: When a user makes changes on a subform (embedded or reusable) and clicks the Save button, the system prompts for an electronic signature for the subform. Scenario 3: If the Pass Through Signature option was enabled on the power form during the electronic signature configuration, the system does not prompt for an electronic signature. Instead, it automatically records the user's information in a signature record. **Note:** The Pass Through Signature option can be used in any of the scenarios described in this table.
On a reusable subform, the Save button is explicitly configured for an electronic signature. The power form is not configured for an electronic signature.	When a user makes changes on the subform and clicks the Save button, the system prompts the user for an electronic signature.
On the power form, the OK button is configured for an electronic signature. On a reusable subform, the electronic signature on the Save button is explicitly turned off.	When a user makes changes on the reusable subform and clicks the Save button, the system determines that the electronic signature is explicitly turned off and does *not* prompt for an electronic signature.
On a power form, the OK button is configured for an electronic signature. On a reusable subform, the electronic signature is explicitly turned off. **Note:** In this scenario, the OK button on the power form has been programmed to invoke the Save button on the subform.	When a user makes changes on both the reusable subform and power form and clicks the OK button on the power form: 1. The system prompts for an electronic signature. 2. The OK button on the power form invokes the Save button on the reusable subform. 3. The system determines that the electronic signature is explicitly turned off and does *not* prompt for another electronic signature.
On a power form, the OK button is configured for an electronic signature. On a reusable subform, the SAVE button is explicitly configured for a different electronic signature. **Note:** In this scenario, the OK button on the power form has been programmed to invoke the Save button on the subform.	When a user makes changes on both the reusable subform and power form and clicks the OK button on the power form: 1. The system prompts for an electronic signature 2. The OK button on the power form invokes the Save button on the reusable subform. 3. The system prompts for a different electronic signature.
On a power form, the OK button is configured for an electronic signature. The reusable subform is explicitly configured to use the same signature as the power form. **Note:** In this scenario, the OK button on the power form has been programmed to invoke the Save button on the subform.	When a user makes changes on both the reusable subform and power form and clicks the OK button on the power form: 1. The system prompts for an electronic signature. 2. The OK button on the power form invokes the Save button on the reusable subform. 3. The system does not prompt for a different electronic signature for the subform; it automatically uses the signature that was entered for the power form.

Configuration	Result
On a power form, the OK button is configured for an electronic signature. On a reusable subform, the electronic signature is explicitly turned off. **Note:** In this scenario, the OK button on the power form has been programmed to invoke the Save button on the subform.	When a user makes changes on both the reusable subform and power form and clicks the OK button on the power form: 1. The system prompts for an electronic signature. 2. The OK button on the power form invokes the Save button on the reusable subform. 3. The system determines that the electronic signature is explicitly turned off.
On a power form, the OK button is *not* configured for an electronic signature. The power form contains two reusable subforms that are explicitly configured for electronic signatures. The OK button on the power form has been programmed to invoke the Save button on both subforms.	When a user makes changes on the power form and reusable subforms and clicks the OK button on the power form: 1. The OK button on the power form invokes the Save button on the first reusable subform and prompts for an electronic signature. 2. The system continues the remaining OK processing and invokes the Save button on the second reusable subform, which prompts for another electronic signature. **Note:** The system does *not* prompt for an electronic signature on the power form.
The electronic signature is configured at the application level. **Note:** If the application contains reusable subforms, you cannot set up electronic signatures at the application level.	When a user makes changes in any of the forms in the application and clicks OK or Save, the system prompts for an electronic signature.

3.1.9 Audit Records, Signature Records, and GUIDs

The system adds a GUID to audit and signature records. You can use the GUID to track the audit and signature records that are associated with the same action. For example, when a user modifies a field configured to trigger an audit and then clicks the OK button configured for a signature approval, the system creates audit and signature records. The system attaches the same GUID to each of these records. You can use this GUID when reviewing and creating reports of audit and signature records that are associated to a particular action.

A GUID can be created in different ways. The following tables shows how a GUID is created in the various database programs:

For Oracle:

Action	GUID	CFRMKEY	CFRPID	CFRUSER
Interactive OW Application	GUID passed from JDB	mkey from JDB	pid from JDB	user from JDB
Interoperability Applications	"OW without GUID"	hostname	program name	login_userID
Direct Table Operations	"Third Party"	hostname	program name	login_userID

For IBM DB2 for LUW (Linux, UNIX, Windows):

Action	GUID	CFRMKEY	CFRPID	CFRUSER
Interactive OW Application	GUID passed from JDB	mkey from JDB	pid from JDB	user from JDB

Action	GUID	CFRMKEY	CFRPID	CFRUSER
Interoperability Applications	"OW without GUID"	"OWNOGUID"	"OWNOGUID"	$USER
Direct Table Operations	"Third Party"	"3RDPARTY"	"3RDPARTY"	$USER

For DB2 for IBM i:

Action	GUID	CFRMKEY	CFRPID	CFRUSER
Interactive OW Application	GUID passed from JDB	mkey from JDB	pid from JDB	user from JDB
Interoperability Applications	"OW without GUID"	"OWNOGUID"	"OWNOGUID"	"OWNOGUID"
Direct Table Operations	"Third Party"	"3RDPARTY"	"3RDPARTY"	"3RDPARTY"

For SQL Server:

Action	GUID	CFRMKEY	CFRPID	CFRUSER
Interactive OW application	GUID passed from JDB	mkey from JDB	pid from JDB	user from JDB
Interoperability applications	"OW without GUID"	host_name()	"OWNOGUID"	user_name()
Direct Table Operations	"Third Party"	host_name()	"3RDPARTY"	user_name()

3.2 Enabling Auditing and Electronic Signatures for a Pathcode

This section provides an overview of the auditing and electronic signature configuration for a pathcode and discusses how to:

- Set up an audit only configuration for a pathcode.
- Set up an audit and electronic signature configuration for a pathcode.
- Modify an existing configuration.
- Review configuration changes.

3.2.1 Understanding the Auditing and Electronic Signature Configuration for a Pathcode

Before you configure auditing and electronic signatures for interactive or batch applications, you must set up an auditing and electronic signature configuration for a pathcode. You can set up an audit-only configuration or a configuration for both auditing and electronic signatures.

When you use the Configuration Application (P9500001) to set up a pathcode for auditing and electronic signatures, you must make sure that you are signed into JD Edwards EnterpriseOne using the pathcode for which you want to enable auditing and electronic signatures. The auditing and electronic signature configuration is recorded in the F9500001 table in the database. A history of all changes to the configuration is recorded in the Security History table (F9312).

Note: Configuring auditing and electronic signatures for more than one pathcode is not supported.

3.2.2 Setting Up an Audit Only Configuration for a Pathcode

An Audit Only configuration is used to record only tables with changes, not electronic signatures.

Enter **P9500001** in the Fast Path.

Note: P9500001 is available on the Microsoft Windows client and the web client.

1. On the Work with Configuration Settings form, click Add.
2. On the Configuration Setting Revisions form, select the Auditing check box.

   The Pathcode field contains the pathcode to which the settings are applied, which is also the pathcode that you are signed in to. This field cannot be modified.
3. Make sure that the Signature check box is *not* selected, and then click OK.
4. On the Enable/Disable Auditing form, in the Password field, enter a password for enabling or disabling auditing.

   The password that you enter here is the password that you must enter if you enable or disable auditing for a particular table.

   See Enabling and Disabling Table Auditing.
5. Click OK.

3.2.3 Setting Up an Audit and Electronic Signature Configuration for a Pathcode

Enter **P9500001** in the Fast Path.

Note: P9500001 is available on the Microsoft Windows client and the web client.

1. On the Work with Configuration Settings form, click Add.
2. On the Configuration Setting Revisions form, select the Signatures check box.

   The Pathcode field displays the pathcode where the settings are applied. This defaults to the pathcode that you are signed in to and cannot be modified.

   The Auditing check box is automatically checked if Signatures is selected. You cannot clear this check box.
3. Select any of the following options, as appropriate:
   – Pass Through User

     Records the user ID of the person using the application, the time of the electronic signature approval, and the machine name.

   – Pre-populate Approver

Automatically uses the user's JD Edwards EnterpriseOne sign-in information for the first and second approval, depending on the configuration for the number of approvers.

– Real Time Notification

  Sends a notification to a specified user if a signature fails.

  Note: You can only use these options if they were enabled at the pathcode level.

  See Enabling Auditing and Electronic Signatures for a Pathcode.

4. In the Default Signature Context field, enter the signature context for the pathcode, and then click OK.

   This text appears on the Signature Capture form unless a specific signature context has been defined at the application level.

5. On the Enable/Disable Auditing form, in the Password field, enter a password for enabling or disabling auditing, and then click OK.

   The password that you enter here is the password that you must enter if you enable or disable auditing for a particular table.

3.2.4 Modifying an Existing Configuration

Enter **P9500001** in the Fast Path.

Note: You can only change a configuration for the pathcode that you are signed into. However, all pathcode configurations may be viewed through this application. You can turn on or turn off the auditing or signature configuration, as well as change the Pass Through User, Real Time Notification, and Pre-populate Approver features for electronic signatures.

---

1. On the Work With Configuration Settings form, select the current pathcode and then click Select.
2. On the Configuration Setting Revisions form, edit the desired parameters noting that:
   - Auditing must be disabled for all tables before disabling auditing at the pathcode level.

     See Enabling and Disabling Table Auditing.
   - Disabling electronic signatures does not turn off the auditing function. After disabling signatures, you can then clear the Auditing check box if you want to turn off auditing for the pathcode.
3. Click OK.
4. On the Enable/Disable Auditing form, if you disabled or enabled auditing, complete the Password field and then click OK.

3.2.5 Reviewing Configuration Changes

Changes made to the configuration can be viewed by using the User Security application (P98OWSEC). This application displays any changes made to the configuration.

3.3 Setting Up Auditing

This section provides an overview of auditing restrictions and discusses how to:

- Configure a table for auditing.
- Check in an audit table.
- Modify an existing audit table.
- Verify and reset audit tables.
- Copy an audit table configuration.

3.3.1 Understanding Auditing Restrictions

Audit columns within audited tables are set to a protected status by TDA to prevent columns from being modified or deleted after the table is enabled for auditing. Therefore, the following restrictions apply:

- You cannot remove columns once auditing is enabled for a table.
- You cannot change the order of the columns once auditing is enabled for a table.

3.3.2 Configuring a Table for Auditing

Enter **P9500003** in the Fast Path, or enter **GH9091** and then select the Audit Table Configuration menu.

1. On the Work with Audit Table Configuration form, click Add.
2. On the Audit Table Setup Director, click Next.
3. On the JD Edwards EnterpriseOne Table Selection form, in the Table Name field, enter the name of the table that you want to audit, and then click Next.

   Alternatively, you can click the Find button and select a table from the list.

   Note: If a table was previously configured for audit, it cannot be selected.

4. On the Audit Table Information form, if necessary, make changes to the default values, and then click Next.

   This action launches TDA in audit mode, which creates audit table specifications locally and adds the audit table to your default project in OMW.

   By default, the Primary Key columns from the original table are populated along with eight more columns, which are mandatory for the audit table.

5. In the Auditable Columns area, select the columns that you want to trigger an audit and move them to the Columns area. You can append new columns to the end of the audit table, but they cannot be inserted between existing columns or reordered.

6. In the Auditable Columns area, select any additional columns that you want recorded in the audit table and move them to the Columns area. Right-click the column and then select Audit Trigger Column.

   This action removes the check mark from this option so that the column will not trigger an audit. Instead, this column will be recorded along with the other columns in the audit table when an audit is triggered.

7. Click Save and then close TDA after all columns have been selected for auditing.
8. On the Work with Audit Table Configurations form, click End.
9. Restart all servers and clients.

Note: After you complete the configuration, you must enable the table for auditing in JD Edwards EnterpriseOne.

See Enabling and Disabling Table Auditing.

3.3.3 Checking In an Audit Table

Checking in an audit table moves the audit table from the local specifications to the check in location specified in the OMW transfer activity rule.

Enter **P9500003** in the Fast Path, or enter **GH9091** and then select the Audit Table Configuration menu.

1. On the Work with Audit Table Configurations form, select the table to check in.
2. From the Row menu, select Check In Table.

3.3.4 Modifying an Existing Audit Table

When you modify an existing audit table, the system checks out the object to your default project in OMW and enables you to append columns to the audit table using TDA.

Enter **P9500003** in the Fast Path or enter **GH9091** and then select the Audit Table Configuration menu.

1. On the Work with Audit Table Configurations form, select the table that you want to modify.
2. From the Row menu, select Set Up Revisions.
3. On the Audit Table Configuration form, click OK.

   The system launches TDA.

4. In TDA, you can move columns from the Auditable Columns area to the Columns area to append columns to the existing audit table.
5. In the Auditable Columns area, select any additional columns that you want recorded in the audit table, and move them to the Columns area. Right-click the column and then select Audit Trigger Column to clear the check mark from this option.

   This action ensures that the column is recorded along with the other columns in the audit table when an audit is triggered. However, this column will not trigger an audit.

6. Click Save, then exit TDA after all columns have been appended for auditing.

7. On the Work with Audit Table Configurations form, from the Row menu, select Check In Table.

3.3.5 Verifying and Resetting Audit Tables

This section explains how to check the configuration for a given table across all environments and databases within the pathcode and explains how to reset the auditing state for the selected audit table if a conflict is detected. Perform this task when the JDE.log or JAS.log file indicates conflicts with audited tables between environments.

Enter **P9500003** in the Fast Path, or enter **GH9091** and then select the Audit Table Configuration menu.

1. On the Work with Audit Table Configurations form, select the audit table that you want to verify.
2. From the Row menu, select Verify.
3. On the Validate and Reset Database Audit Components form, enter a database password in the Password field, and then click OK.

   If the audit configuration is valid, the system displays the Work with Audit Table Configurations with an informational message, "Database Audit Components valid."
4. If the audit configuration is not valid, the following error is returned on the Validate and Reset Database Audit Components form: Error: Database Audit Component States Differ. From the Form menu, select Reset Audit Flag to reset database audit components.
5. Click OK on the message box reminding you to bounce servers and clients.
6. Click Find to refresh the Work with Audit Table Configurations form. Verify that the Audit table shows the correct state: Enabled or Disabled.

3.3.6 Copying an Audit Table Configuration

If the audited tables reside in a data source shared between environments within the same pathcode, use the copy table function to add the audit table configuration to each environment.

JD Edwards EnterpriseOne requires that you copy the audited table configuration to the other environments prior to enabling auditing. Auditing the table only in one environment causes the other environments to fail in the application that uses that particular table, rolling back the transaction. The JDE.log and JAS.log files indicate conflicts with audited tables between environments.

1. Sign in to the environment where the audit configuration needs to be set up, and then access the Work with Audit Table Configurations form.
2. On the Work with Audit Table Configurations form, from the Form menu, select Copy Audit Table.
3. On the Copy Audit Table Information form, find and select the table to be copied.
4. Verify that the selected audit table is in the configuration tree.

> **Note:** JD Edwards EnterpriseOne displays only the tables that have been set up for a different environment, checked in to the shared data source, and are missing from the audit configuration for the signed-in environment.

3.4 Enabling and Disabling Table Auditing

This section discusses how to:

- Enable auditing for a table.
- Disable table auditing for an Oracle, SQL Server, or DB2 for IBM i database.
- Disable table auditing for a IBM DB2 for LUW (Linux, UNIX, Windows) database.

3.4.1 Enabling Auditing for a Table

This section describes how to enable auditing for the desired table after auditing has been enabled for the pathcode.

> **Note:** Make sure that all users are signed off the system and no batch jobs are running on the server.

Enter **P9500003** in the Fast Path, or enter **GH9091** and then select the Audit Table Configuration menu.

1. On the Work with Audit Table Configurations form, select the audit table.
2. From the Row menu, select Enable DB Auditing.
3. On the Enable/Disable Database Auditing Components form, enter the database password in the Password field.
4. Click OK to enable auditing for the table.

   The application reminds you that all users must be signed out of JD Edwards EnterpriseOne and all enterprise and JAS servers must be restarted.

   > **Note:** You only need to restart all servers and clients once, after the configuration for all tables and objects are complete.

5. Click OK.

   The audit table record in the Work with Audit Table Configurations tree indicates that auditing is enabled for the table.

3.4.2 Disabling Table Auditing for an Oracle, SQL Server, or DB2 for IBM i Database

Before performing this task, make sure that all users are logged off the system and that no batch jobs are running on the server.

> **Note:** Disabling auditing for a table does not remove the table from the audit list.

1. On the Work with Audit Table Configurations form, select the audit table.

2. From the Row menu, select Disable DB Auditing.
3. On the Enable/Disable Database Auditing Components form, enter the database password in the Password field.
4. Click OK to disable auditing for the table.

   The system displays a message recommending that you bounce all servers and clients.

   > **Note:** You have to bounce all servers and clients only once, after the configuration of all tables and objects is complete.

5. Click OK.

   The Work with Audit Table Configurations form displays the Audit table record in the tree control and indicates that auditing is disabled.

3.4.3 Disabling Table Auditing for a IBM DB2 for LUW (Linux, UNIX, Windows) Database

Before performing this task, make sure that all users are logged off the system and that no batch jobs are running on the server.

> **Note:** Disabling auditing for a table does not remove the table from the audit list.

1. From the Start menu, select IBM DB2, Command Window to start a DB2 Command Window.
2. Sign in to the database with a user who has a DBADM privilege.
3. Collect Table Space information for the Specified table with this SQL Command:

```
SELECT TBSPACE, INDEX_TBSPACE, LONG_TBSPACE  FROM

SYSCAT.TABLES where TABNAME = ERPTable (with '') and

TABSCHEMA=schema (with'')
```

4. Write down the table space names for table, index, and long data, respectively.
5. Create the temporary table with this SQL command:

```
CREATE TABLE schema.ERPTable_temp LIKE schema.ERPTable (VIEW) IN tbspace INDEX
IN⇒
 index_tbspace LONG IN long_tbspace NOT LOGGED INITIALLY
```

   > **Note:** The table space names were collected from step 4: tbspace for table, index_tbspace for index, and long_tbspace for long data.
   >
   > Run the SQL command in step 3 against the new temporary table to ensure that it has been created in correct table spaces.

6. Copy data from the view to the temporary table with this SQL command:

```
INSERT INTO schema.ERPTable_temp SELECT * FROM schema.ERPTable (VIEW)
```

Note: This process may take a few minutes to complete depending on the size of the table.

---

7. To ensure that all data was copied, run the following SQL commands against the view and the temporary table:

   ```
   SELECT COUNT(*) FROM schema.ERPTable_temp

   SELECT COUNT(*) FROM schema.ERPTable
   ```

   The results of running both commands should be same.

8. Use this SQL command to drop the view:

   ```
   DROP VIEW schema.ERPTable
   ```

9. Use this SQL command to drop the ADT table:

   ```
   DROP TABLE schema.ERPTable_ADT
   ```

10. Rename temporary table with the original JD Edwards EnterpriseOne table name:

    ```
    RENAME schema.ERPTable_temp TO ERPTable
    ```

11. Use this SQL command to grant privileges to PUBLIC for the new table:

    ```
    GRANT SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX, REFERENCES ON TABLE⇒
     schema.ERPTable TO PUBLIC
    ```

12. Sign in to JD Edwards EnterpriseOne client and then enter **OMW** in the Fast Path to access OMW.
13. Use Table Design Toolset to generate indices for this table.
14. Run "Verify and Reset" using the Work With Audit Table Configuration form.

    See Verifying and Resetting Audit Tables.

3.5 Setting Up Reason Codes

You must set up reason codes before you configure electronic signatures. This section discusses how to:

- Add a reason code.
- Change the status of a reason code.
- Copy a reason code.

3.5.1 Adding a Reason Code

Enter **P9500002** in the Fast Path.

1. On the Work with Reason Codes form, click Add.
2. On Reason Code Revisions, in the Reason Code field, enter a number for the reason code.
3. In the Language field, enter the language code for the reason code.
4. In the Reason Code Description field, enter a description for the reason code.
5. Select the Active or Inactive option, as appropriate, and then click OK.

Note: A warning message appears to verify that the spelling of the reason code description is correct. Reason codes cannot be deleted.

3.5.2 Changing the Status of a Reason Code

Changing the status of a reason code changes it from active to inactive or vice versa.

1. On the Work with Reason Code form, select the desired reason code.
2. Under the Row menu, select Change Status.

Note: You cannot change the status of a reason code if it is the default reason code in an electronic signature configuration.

3.5.3 Copying a Reason Code

Copying a reason code copies all fields except the reason code number.

1. Select the desired reason code to copy and then click Copy.
2. Change the desired field and provide a new reason code number.
3. Click OK.

3.6 Setting Up Electronic Signatures

Use the Object and Table Configuration (P9500003) application to set up interactive or batch application to require electronic signatures. This section lists a prerequisite and discusses how to:

- Set up an electronic signature configuration for an interactive application.
- Set up an electronic signature configuration for a batch application.
- Modify an electronic signature configuration.
- Delete an electronic signature configuration.

3.6.1 Prerequisite

You must define reason codes before you set up electronic signatures.

See Setting Up Reason Codes.

3.6.2 Setting Up Electronic Signatures for an Interactive Application

Enter **GH9091** in the Fast Path, and then select Interactive and Batch Signature Configuration.

1. On the Work with Object Signature Configurations form, click Add.
2. On the Interactive and Batch Applications Signature Setup Director, select the Interactive Application/Signature Setup option, and then click Next.
3. On the Interactive Application Selection form, select the desired application, and then click Next.

Note: Objects with a check mark next to them have already been configured.

---

4. On the Interactive Forms Search and Select form, select a form. If all forms for an application are desired, select the Application Level check box.

Note: Applications with reusable subforms cannot be configured at the application level.

---

5. Click Next.

Note: If there is a padlock button to the left of the Form Name, this form already has the OK and Delete events set up and cannot be selected. Instead, this form must be selected from the Work with Signature Object Configurations form. Select Setup Revisions from the Row menu to revise an existing configuration.

A plus sign (+) indicates that an OK event has already been set up for this form.

A minus sign (-) indicates that a Delete event has already been set up for this form.

---

6. On the Interactive Application Signature Information form, in the Events area, select OK, Delete, or both to indicate the function that requires a signature.
7. For a reusable subform, you can select the "Exclude Subform from Signature Configuration" option.

   This option is available only for reusable subforms. If the hosting power form is configured for an electronic signature, you can select this option if you do not want the system to prompt for a second electronic signature on the reusable subform.

   See Electronic Signatures for Power Forms and Subforms.
8. Select one of these options:
   – Pre-populate Approver

     Select this option if you want the authorization fields to automatically use the session's user information.

   – Pass Through User

     Select this option if no signature prompting is necessary when the application is run. The signature table is updated with the signed on user's information.

Note: You can select either the Pre-populate Approver or Pass Through User option. They cannot be selected together.

If the Pass Through User is on, the number of signatures is forced to one and the role is disabled.

---

9. Enter the number of signatures required to approve the transaction: **1** or **2.**

10. In the Default Reason Code field, enter the default reason code for the signature approval.
11. In the Role field, enter the role of the approver.
12. In the Real Time Notification field, enter the user ID or email address of the user to be notified if the electronic signature fails.

    The user is notified after four failed attempts.
13. In the Signature Context field, enter a text message that you want to display on the Signature Approval form, and then click Next.
14. On the Work with Object Signature Configurations form, click End.

3.6.3 Setting Up Electronic Signatures for a Batch Application

Enter **GH9091** in the Fast Path, and then select Interactive and Batch Signature Configuration.

1. On the Work with Configurations form, select Object Setup from the Form menu.
2. On the Work with Object Signature Configurations form, click Add.
3. On the Interactive and Batch Applications Signature Setup Director, select the Batch Application / Signature Setup option, and then click Next.
4. On the Batch Application Selection form, select the desired batch application, and then click Next.

    Note: Objects with a check mark next to them have already been configured.

5. On the Batch Application Signature Information form, select one of these options:
    - Pre-populate Approver.

      Select this option if you want the authorization fields to automatically use the session's user information.
    - Pass Through User.

      Select this option if no signature prompting is necessary when the application is run. The signature table is updated with the signed on user's information.

    Note: Pre-populate Approver and Pass Through User cannot be selected at the same time.

    If the Pass Through User is on, the number of signatures is forced to one and the role is disabled.

6. Enter the number of signatures required to approve the transaction: **1** or **2.**
7. In the Default Reason Code field, enter the default reason code for the signature approval.
8. If a role is required, enter the role of the approver in the Role field.
9. In the Real Time Notification field, enter the user ID or email address of the user to be notified after four failed signature attempts.

10. In the Signature Context field, enter a text message that you want to display on the Signature Approval form, and then click Next.
11. On the Work with Object Signature Configurations form, click End.

3.6.4 Modifying an Electronic Signature Configuration

Enter **GH9091** in the Fast Path, and then select Interactive and Batch Signature Configuration.

1. On the Work with Object Signature Configurations form, select the interactive application, event, or batch application to modify.
2. Select Setup Revisions from the Row menu.

   Depending on the type of application that you are modifying, the system displays either the Interactive Application Signature Information form or Batch Application Information form
3. Make changes and then click OK.
4. Verify that the properties have changed for the interactive application, event, or batch application.

Note: All servers and clients need to be restarted once after the auditing and electronic signature configuration of tables and applications (interactive or batch) is complete. Any changes that you make to the auditing or electronic signature configuration will require services to be restarted on all servers and clients.

3.6.5 Deleting an Electronic Signature Configuration

Enter **GH9091** in the Fast Path, and then select Interactive and Batch Signature Configuration.

1. On the Work with Object Signature Configurations form, click Find and then select the interactive application event or batch application to delete.
2. Click Delete.
3. Verify that the interactive application, event, or batch application has been removed from the tree on the Work with Object Signature Configuration form.

4

Working with Auditing and Electronic Signature Approvals

This chapter contains the following topics:

4.1 Entering an Electronic Signature Approval

This section provides an overview of the Signature Approval form and discusses how to use the Signature Approval form.

4.1.1 Understanding the Signature Approval Form

The Signature Approval form is used to require an electronic signature approval before an interactive or batch application can continue. When attempting to save or delete information in interactive applications or run batch applications that have been configured to require an approval, users are prompted for an approver's user ID and password. After an approver enters this information, the system verifies the credentials and then records it in a log file.

A signature consists of an approver's ID, password, and reason code. If the signature is valid, JD Edwards EnterpriseOne records who approved the change, the role, the reason, the application used, and the date and time in the Signature table (F9500006). An approver has four attempts to enter a valid user ID and password. If the signature is not valid, an error message appears and the user has three more attempts to enter the correct information. If the fourth signature attempt fails, a failed signature record is written to the F9500006 table.

The Signature Approval form can require up to two signature approvals. The second approver has the option to view the previous approver's information. In addition, users can attach media objects to electronic signature approvals.

4.1.2 Using the Signature Approval Form

When an application has been configured for signature approval, it launches the Signature Approval form before data can be processed or deleted.

1. On the Signature Approval form, complete the User ID and Password fields.
2. Select a reason code in the Reason Code field.
3. If needed, enter additional information for the transaction in the Comment field.

4. To add a media object attachment to the approval, click the attachment button.
5. If two signature approvals are required, the second approver can view the previous approver's information by selecting Previous Info from the Form menu.

   After reviewing the information, the second approver can click Cancel to return to the current approval form.
6. Click OK to validate the approval and allow the application to continue.

   Clicking Cancel on this form stops the transaction from processing and does not log an approval.

4.2 Removing Unmatched Audit Records

Before you review audit and signature records, you can run the R9500005A (Delete Unmatched Audit Log Records) batch application to remove any unmatched (or invalid) audit records. Audit records in the F9500003 and F9500004 tables are invalid if they do not have associated audit records in the corresponding audit table. The system can create unmatched records due to a system failure, an error, or other circumstance.

This section discusses how to run the R9500005A batch application to remove unmatched audit records.

4.2.1 Running the R9500005A Batch Application to Remove Unmatched Audit Records

Enter **BV** in the Fast Path to access the Batch Versions applications.

1. On the Work With Batch Versions - Available Versions form, enter **R9500005A** in the Batch Application field and click the Find button.
2. Select the Delete Unmatched Audit Log Records batch application and then select the Row menu, Processing Options.
3. On the Processing Options form, enter the appropriate processing option value in the Mode for Deletion field and then click the OK button:
   - Enter 0 for proof mode.

     This processing option enables you to view a report of the invalid records that the batch application will delete when run in final mode.
   - Enter 1 for final mode, which deletes all unmatched records.

     This processing option removes invalid records and generates a report that lists the invalid records that were removed.
   - Enter 2 for final mode, which deletes unmatched records but retains records if an audit table could not be opened.

     This processing option removes invalid records on a table unless the table associated with the invalid records has been taken offline; then the system will not delete the invalid records.
4. On the Work With Batch Versions - Available Versions form, click the Select button.
5. On the Batch Versions - Version Prompting form, click the Submit button.
6. On the Printer Selection form, click the OK button.
7. On the Work With Batch Versions - Available Versions form, select the Form menu, Submitted Jobs.

8. On Batch Versions - Submitted Job Search, to review the results of the batch job, select the job in the grid and then select the Row menu, View PDF.

4.3 Viewing Audit and Electronic Signature Records

The View Audit/Signature Information application (P9500005) enables you to view detailed records of signature and audit information. This application is available on Microsoft Windows client or the web client.

This section lists prerequisites and discusses how to:

- View audit information.
- View signature information.
- View audit and signature information together.

4.3.1 Prerequisites

Before you can view audit and electronic signature records:

- Specifications for the audit shadow table must exist on the enterprise server. Therefore, after you configure auditing for a table, you must build and deploy a package to the enterprise server.
- To view audit information on the web client, you must build and deploy a package that contains all audit table specifications to the enterprise server.

4.3.2 Viewing Audit Information

Enter **GH9091** in the Fast Path, and select the View Audit and Signature Information menu.

1. On the Work with Audit and Signature Information form, click the Audit Information link.
2. On the Work with Audit Information form, complete the appropriate fields to search for the desired information, and then click Find:
   - Table Name

     Search by the name of the table that was audited (Fxxxx).
   - Audit Table Name

     Search by the name of the audit table (Axxxx).
   - Date Updated

     Search by the date the audit record was created.
3. Select the Matched Audit Records Only check box to refine the records so that the system only displays the audit records (from the F9500003 and F9500004 tables) that match the audit records in the corresponding audit table (Axxxx).

   This filters unmatched audit records that can be created due to a system failure or other circumstance.

   Note: You can also run a batch application that deletes all unmatched records from the table.

   See Removing Unmatched Audit Records.

4. Click Find.
5. Select the desired record and click Select.
6. Select one or a combination of the following options to refine the search, and then click OK:
   - Unique Identifier
   - Table Name
   - Audit Table Name
   - Data Source
   - User ID
   - Address Number
   - Full Name
   - Date Updated
   - Time Last Updated
   - Program ID
   - Machine Key
   - IP Address
7. To view a report of the audit information in PDF, select the Report Menu, Print Data.

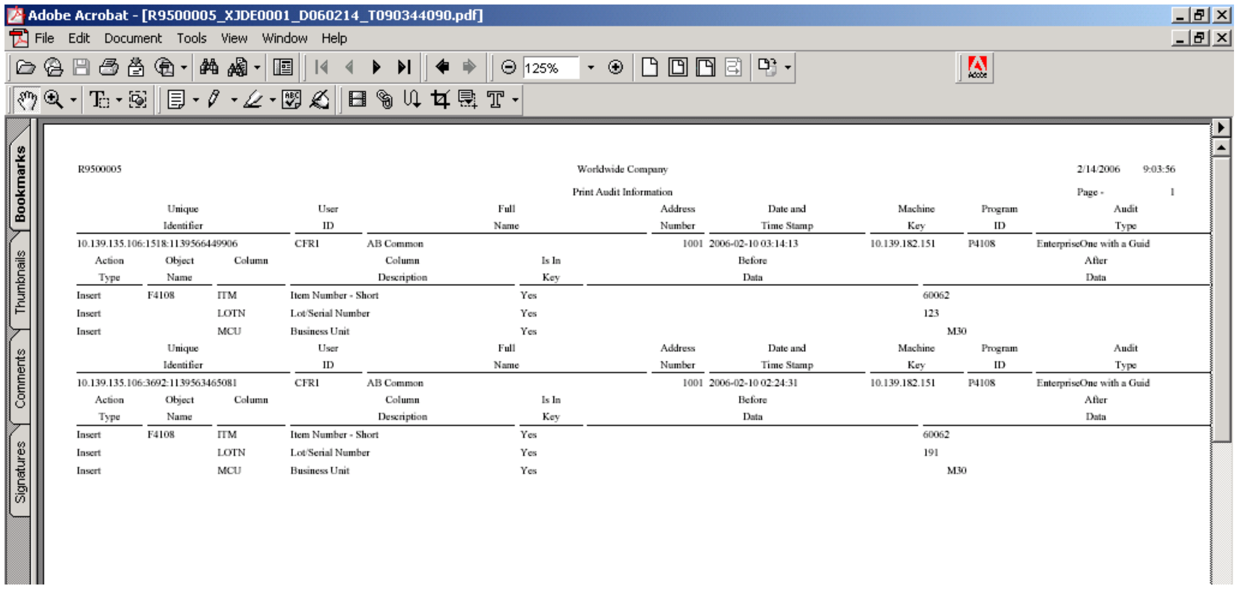
   Note: If multiple users will be using this application, the Temporary Audit Table (F9500005) must be mapped locally.

8. To view the signature information that corresponds to the audit record, on the Work with Audit Detail Information form, select the Row menu, Link to Signature Info.
9. On the Work with Signature Information form, to access a blank Search Audit Info form, select the Form menu, Input Search Info.
10. On the Search Audit Info form, enter the desired search parameters in the appropriate fields, select the corresponding check box, and then click OK.

4.3.2.1 Example of Audit Information Report

The following example shows the results of running the Audit Information report:

Figure 4–1 Audit Information report


Adobe Acrobat - [R9500005_XJDE0001_D060214_T090344090.pdf]
File Edit Document Tools View Window Help
125%
Bookmarks
Thumbnails
Comments
Signatures

```
R9500005                                        Worldwide Company                                          2/14/2006     9:03:56
                                                Print Audit Information                                    Page -              1
            Unique              User                 Full                 Address      Date and       Machine      Program        Audit
          Identifier             ID                  Name                  Number     Time Stamp        Key           ID           Type
10.139.135.106:1518:1139566449906  CFR1   AB Common                          1001  2006-02-10 03:14:13  10.139.182.151  P4108   EnterpriseOne with a Guid
   Action      Object      Column            Column               Is In                   Before                                   After
    Type        Name                        Description             Key                     Data                                    Data
Insert      F4108       ITM       Item Number - Short            Yes                                          60062
Insert                  LOTN      Lot/Serial Number              Yes                                          123
Insert                  MCU       Business Unit                  Yes                                              M30
            Unique              User                 Full                 Address      Date and       Machine      Program        Audit
          Identifier             ID                  Name                  Number     Time Stamp        Key           ID           Type
10.139.135.106:3692:1139563465081  CFR1   AB Common                          1001  2006-02-10 02:24:31  10.139.182.151  P4108   EnterpriseOne with a Guid
   Action      Object      Column            Column               Is In                   Before                                   After
    Type        Name                        Description             Key                     Data                                    Data
Insert      F4108       ITM       Item Number - Short            Yes                                          60062
Insert                  LOTN      Lot/Serial Number              Yes                                          191
Insert                  MCU       Business Unit                  Yes                                              M30
```

4.3.3 Viewing Signature Information

Enter **GH9091** in the Fast Path, and select the View Audit and Signature Information menu.

1. On the Work with Audit and Signature Information form, click the Signature Information link.
2. On the Work with Signature Information form, complete any of these fields and then click Find to search for signature records:
   - Application Name
   - Version
   - Form Name
   - Subform Name
   - Event

     Use this field to search by the event executed when the signature was captured (1 = OK, 2 = Delete)
   - Signature Successful

     Use this field to search by the success or failure value of the approval (0 = Success, 1 = Failure).
3. Select the desired signature record and click Select.

   Alternatively, from the Row menu select More Info to see additional details about the signature.
4. On the More Signature Information form, select the Form menu, Link to Audit Info.
5. On the Work with Audit Information form, to print the signature information in PDF format, select the Report menu, Print Data

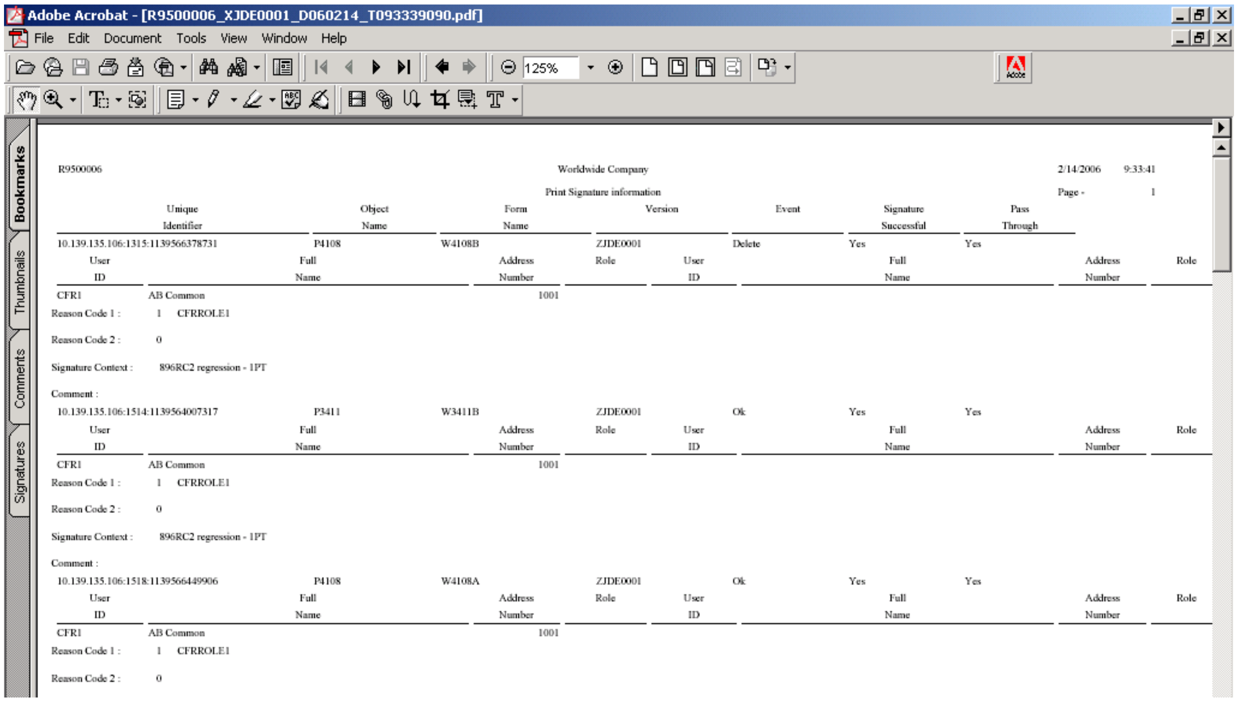
---

Note: If multiple users will be using this application, the Temporary Audit Table (F9500005) must be mapped locally.

4.3.3.1 Example of a Signature Information Report

The following example shows the results of running the Signature Information report:

Figure 4–2 Signature Information report


Adobe Acrobat - [R9500006_XJDE0001_D060214_T093339090.pdf]
File Edit Document Tools View Window Help
125%
Bookmarks
Thumbnails
Comments
Signatures

```
R9500006                                Worldwide Company                                          2/14/2006    9:33:41
                                    Print Signature information                                    Page -            1
            Unique                  Object          Form          Version       Event     Signature     Pass
          Identifier                 Name           Name                                  Successful   Through
10.139.135.106:1315:1139566378731   P4108         W4108B        ZJDE0001        Delete    Yes          Yes
     User                 Full                   Address      Role      User                  Full                Address   Role
      ID                  Name                   Number                  ID                   Name                Number
CFR1          AB Common                               1001
Reason Code 1 :     1    CFRROLE1

Reason Code 2 :     0

Signature Context :    896RC2 regression - 1PT

Comment :
10.139.135.106:1514:1139564007317   P3411         W3411B        ZJDE0001        Ok        Yes          Yes
     User                 Full                   Address      Role      User                  Full                Address   Role
      ID                  Name                   Number                  ID                   Name                Number
CFR1          AB Common                               1001
Reason Code 1 :     1    CFRROLE1

Reason Code 2 :     0

Signature Context :    896RC2 regression - 1PT

Comment :
10.139.135.106:1518:1139566449906   P4108         W4108A        ZJDE0001        Ok        Yes          Yes
     User                 Full                   Address      Role      User                  Full                Address   Role
      ID                  Name                   Number                  ID                   Name                Number
CFR1          AB Common                               1001
Reason Code 1 :     1    CFRROLE1

Reason Code 2 :     0
```

4.3.4 Viewing Audit and Signature Information Together

Enter **GH9091** in the Fast Path, and select the View Audit and Signature Information menu.

1. Click the Signature and Audit Information link.
2. On the Work with Signature and Auditing Information form, complete any of these fields and then click Find to search for signature and audit records:
   - Application Name
   - Version
   - Form Name
   - Subform Name
   - Event. (Search by the event executed when the signature was captured [1 = OK, 2 = Delete].)
   - Table Name. (Search by the name of the audited table [Fxxxx].)
   - Audit Table Name. (Search by the name of the Audit table [Axxxx].)
   - Signature Successful. (Search by the success or failure value of the approval [0 = Successful, 1 = Failure].)

3. On the Work with Signature and Auditing Information form, select the desired record and click Select.
4. On the Search Audit Info form, select one or a combination of the following options to refine the search, and then click OK:
   – Unique Identifier
   – Table Name
   – Audit Table Name
   – Data Source
   – User ID
   – Address Number
   – Full Name
   – Date Updated
   – Time Last Updated
   – Program ID
   – Machine Key
   – IP Address
5. To view a report of the audit information in PDF, select the Report menu, Print Data.
6. To view a report that contains both signature and audit information, select the Report menu, Print Information.

A Configuring Auditing for Interoperability Transactions

This appendix contains the following topics:

A.1 Overview

Audited columns in a table can be updated by transactions that originate outside of JD Edwards EnterpriseOne through an interoperability model. When this occurs, JD Edwards EnterpriseOne creates an audit record for the transaction, but the system records only a portion of the audit information, such as the audited column, before and after values, and recorded columns. The audit information will not include a GUID, application ID, workstation name, or IP address, unless you configure the interoperability model to pass this data to the audit record.

Oracle provides audit APIs that enable the transfer of audit information from these interoperability types:

- Business services
- Java connector
- COM connector
- WSG or XPI

This table describes the audit record columns, the API used to pass the data to the columns, and the values that you enter as parameters for the API:

Audit Record Column	API	Value for API Parameter
GUID	setGUID	Enter an available GUID. If you do not enter a GUID, the API generates a new GUID.

Audit Record Column	API	Value for API Parameter
ApplicationID	setApplicationID	Enter the application ID. If you do not enter an application ID, the API passes "INTEROP" for this value.
WorkstationName	setWorkstationName	Enter the workstation name. If you do not enter a workstation name, the API passes the local workstation name for this value.
IPAddress	setIPAddress	Enter the IP address. If you do not enter an IP address, the API passes the IP address of the local workstation.

Oracle provides two sets of these APIs to pass audit information from interoperability transactions. The set of APIs used depends on the interoperability model being employed, as described in this table:

Interoperability Model	API Set	Platforms
WSG or XPI, Java connector, and business services	JAVA APIs	JD Edwards EnterpriseOne supported platforms
COM Connector	C APIs	WIN 32 platforms

For more information about the APIs, see the *JD Edwards EnterpriseOne Tools Release 8.98 API Reference Guide* on My Oracle Support Web site.

A.2 Reference Implementations for Auditing

The instructions in this chapter use reference implementations to describe how to configure interoperability models to pass audit information. Reference implementations are samples of fully functional code that you can use as an example for setting up or testing these interoperability implementations:

- Business services
- Java connector
- COM connector
- WSG or XPI

Note: Each section in this chapter includes the location of the resources and files for the reference implementations.

A.3 Configuring a Published Business Service to Pass Audit Information

This section provides an overview of configuring a published business service to pass audit information and discusses how to:

- Configure RI_AuditAddressBookManager to pass audit information.
- Configure RI_AuditInsertABStagingManager to pass audit information.

A.3.1 Overview

You can configure a published business service to pass audit information to an audit record. To do so, you must create a new published business service class that extends from an existing published business service class—the class that invokes a business function or database operation.

A.3.1.1 Using Reference Implementations to Test a Business Services Auditing Configuration

The tasks in this section use reference implementations to describe how to configure business services to pass audit information. You can use the reference implementations to configure and test an auditing configuration, which includes these high level steps:

1. In Object Management Workbench (OMW), add the reference implementations to a project and then follow the instructions in this section that describe how to configure the reference implementations for auditing.
2. Add the references implementations to a package and deploy them in a test environment.

   See "Working with Packages for Business Services" in the *JD Edwards EnterpriseOne Tools Package Management Guide*.
3. Run the reference implementations.

Note: Reference implementations are intended for test purposes only. Do not use them in a runtime environment.

A.3.2 Configuring RI_AuditAddressBookManager to Pass Audit Information

The RI_AddressBookManager reference implementation is an example of a published business service that adds an address book record to JD Edwards EnterpriseOne. This section describes how to use the RI_AuditAddressBookManager reference implementation to extend the RI_AddressBookManager so that audit information is passed to an audit record when an address book record is added to the system.

For more information about the RI_AddressBookManager reference implementation, see the *JD Edwards EnterpriseOne Tools 8.98 Interoperability Reference Implementations Guide* on My Oracle Support.

Oracle provides the JPR95001 – RI_AuditAddressBookManager reference implementation, which you can use to extend the RI_AddressBoookManager published business service to pass audit information.

This table describes the components of JPR95001 –RI_AuditAddressBookManager and the location of these components in the JD Edwards EnterpriseOne installation directory:

Reference Implementation Component	Description	Location
RI_AuditAddressBookManager	A published business service class that extends the AddressBookManager reference implementation so that it inserts an address book record with audit information.	B9\STAGINGA\java\source\oracle \e1\bssv\JPR95001

Reference Implementation Component	Description	Location
auditAddAddressBook	A method in the RI_ AuditAddressBookManager that is used to insert an address book record with audit information. This method uses the addAddressBook method of the RI_ AddressBookManager class to insert the address book record.	Same as previous.
RI_AuditAddAddressBook	The value object class that contains get and set methods for the GUID, application ID, workstation name, and IP address.	B9\STAGINGA\java\source\oracle \e1\bssv\JPR95001\valueobject

To configure the RI_AuditAddressBookManager reference implementation to pass audit information:

1. Create a new value object class that extends the value object class of the existing published business service class.

   For example:

   ```
   public class RI_AuditAddAddressBook extends RI_AddAddressBook implements⇒
    Serializable
   ```

   In this example, RI_AuditAddAddressBook is the new value object class.

   RI_AddAddressBook is the value object class of the RI_AddressBookManager published business service class.

2. In the RI_AuditAddAddressBook value object, add fields for the GUID, application ID, workstation name, and IP address audit information.

   Generate accessors (get and set methods) for these fields.

3. Create a new published business service class to extend the existing published business service class.

   For example:

   ```
   public class RI_AuditAddressBookManager extends RI_AddressBookManager{
   }
   ```

   In this example, RI_AuditAddressBookManager is the new published business service class that is extending the existing published business service, RI_ AddressBookManager.

   RI_AddressBookManager adds an address book record. RI_ AuditAddressBookManager is the class that uses the RI_AddressBookManager class to insert an address book record into the address book table with audit information.

4. Add the new published business service method that adds an address book record with audit information.

   This new method uses the published method of the existing published business service class to add the address book record. Make the new value object class, which was created to pass the audit information, a parameter of this method.

   For example:

   ```
           public RI_ConfirmAddAddressBook
          auditAddAddressBook(RI_AuditAddAddressBook vo) throws
   ```

```
      BusinessServiceException
```

In this example, auditAddAddressBook is the method that is used to add an address book record with the audit information. This method uses the addAddressBook method of the RI_AddressBookManager class to add the address book record.

The new value object, RI_AuditAddAddressBook, is passed as a parameter of the auditAddAddressBook method. The return value of the auditAddAddressBook method is RI_ConfirmAddAddressBook, which indicates whether the address book record was added successfully or errors occurred.

5. In the new published business service class, RI_AuditAddressBookManager, create the context. This is where all the audit information is set.

   For example:

```
context = startPublishedMethod(context, "addAddressBook",vo);
```

6. Add code that checks whether a GUID is provided in the request, and if not, generates a GUID in the class.

   For example:

```
if(vo.getGuid()== null || vo.getGuid().equals("")) {
                guid = UniqueKeyGenerator.getNextGuid();
            }
     else {
                   guid = vo.getGuid();
            }
```

7. Set the audit fields in the context object.

   For example:

```
context.setGUID(guid);
context.setApplicationID(vo.getApplicationID());
      context.setWorkstationName(vo.getWorkstationName());
      context.setIPAddress(vo.getIpaddress());
```

8. Call the published business service method in the existing business service class and return the response.

   For example:

```
     RI_ConfirmAddAddressBook confirmVO = this.addAddressBook(context,
     connection, vo);
    finishPublishedMethod(context, "auditAddAddressBook");
    return confirmVO;
```

   In this example, the RI_AuditAddressBookManager is calling the addAddressBook method of the RI_AddressBookManager to add an address book record with audit information, which is set in the context object. The return value confirmVO indicates whether the record was added successfully or errors occurred.

A.3.3 Configuring RI_AuditInsertABStagingManager to Pass Audit Information

The RI_AddressBookStagingManager reference implementation is an example of a published business service that adds an address book record directly to the interoperability table.

This section describes how to configure the RI_AuditInsertABStagingManager class to extend RI_AddressBookStagingManager, so that audit information is passed with the address book record that is added to the interoperability table.

For more information about the AddressBookStagingManager reference implementation, see the *JD Edwards EnterpriseOne Tools Interoperability Reference Implementations Guide.*

Oracle provides the JPR95002 – RI_AuditInsertABStagingManager reference implementation, which you can use to extend the RI_AddressBoookStagingManager published business service to pass audit information.

This table describes the components of JPR95002 –RI_AuditInsertABStagingManager and the location of these components in the JD Edwards EnterpriseOne installation directory:

Reference Implementation	**Description**	**Location**
RI_AuditInsertABStagingManager	A published business service class that extends the AddressBookStagingManager reference implementation so that it inserts a record with audit information into the interoperability table.	B9\STAGINGA\java\source\oracle\e1\bssv\JPR95002
auditInsertAddressBookStaging	A method in the RI_ AuditInsertABStagingManager that is used to insert an address book record with audit information into the interoperability table. This method uses the insertAddressBookStaging method of the RI_AddressBookStagingManager class to insert the address book record.	Same as previous.
RI_ AuditInsertAddressBookStaging	The value object class that contains get and set methods for the GUID, application ID, workstation name, and IP address.	B9\STAGINGA\java\source\oracle\e1\bssv\JPR95002\valueobject

To configure RI_AuditInsertABStagingManager to pass audit information:

1. Create a new value object that extends the value object of the existing published business service, which in this case is RI_AddressBookStagingManager.

   For example:

   ```
   public class RI_AuditInsertAddressBookStaging extends
   RI_InsertAddressBookStaging implements Serializable
   ```

   RI_AuditInsertAddressBookStaging is the new value object.

   RI_InsertAddressBookStaging is the value object of the RI_ AddressBookStagingManager published business service.

2. In the RI_AuditInsertAddressBookStaging value object, add fields for the GUID, application ID, workstation name, and IP address.

   Generate accessors (get and set methods) for these fields.

3. Create a new published business service that extends the existing published business service.

   For example:

   ```
   public class RI_AuditInsertABStagingManager extends
   ```

```
RI_AddressBookStagingManager {
}
```

RI_AddressBookStagingManager is the existing published business service that inserts an address book record into the interoperability table.

RI_AuditInsertABStagingManager is the new published business service that uses the RI_AddressBookStagingManager class to insert an address book record with audit information into the interoperability table.

4. Add a new published method that inserts an address book record with audit information into the interoperability table.

   This method uses the published method of the parent published business service to insert the address book record. Make the newly created value object a parameter of this method.

   For example:

```
public RI_ ConfirmInsertAddressBookStaging
     auditInsertAddressBookStaging(RI_AuditInsertAddressBookStaging vo) throws
     BusinessServiceException
```

   In this example, auditInsertAddressBookStaging is the method that is used to insert an address book record with audit information into the interoperability table. This method uses the insertAddressBookStaging method in the RI_ AddressBookStagingManager to insert the address book record.

   The new value object, RI_AuditInsertAddressBookStaging, is passed as a parameter. The return value of the auditInsertAddressBookStaging method is RI_ ConfirmInsertAddressBookStaging – which indicates whether the address book record was successfully inserted into the interoperability table or errors occurred.

5. In the new published business service, RI_AuditInsertABStagingManager, create the context.

   This is where you set the audit information.

   For example:

```
context = startPublishedMethod(context, "auditInsertAddressBookStaging",
            vo);
```

6. Add code that checks whether a GUID is provided in the request, and if not, generates a GUID in the class.

   For example:

```
if(vo.getGuid()== null || vo.getGuid().equals("")) {
                guid = UniqueKeyGenerator.getNextGuid();;
            }
            else {
                guid = vo.getGuid();
            }
```

7. Set the audit fields in the context object.

   For example:

```
context.setGUID(guid);
context.setApplicationID(vo.getApplicationID());
      context.setWorkstationName(vo.getWorkstationName());
      context.setIPAddress(vo.getIpaddress());
```

8. Call the method of the existing business service class and return the response.

   For example:

```
        RI_ConfirmInsertAddressBookStaging confirmVO =
        this.insertAddressBookStaging(context, connection, vo);
       finishPublishedMethod(context, "auditInsertAddressBookStaging");
       return confirm;
```

A.4 Configuring a Java Connector to Pass Audit Information

This section provides an overview and discusses:

- How to reuse audit data for multiple business functions.
- Reference implementation resources.
- How to run the Java connector reference implementation to test auditing.

A.4.1 Overview

This section describes how to configure and run the Purchase Order with Audit Feature reference implementation to pass audit information. Refer to the sample code in this section when you configure auditing for a transaction that occurs through a Java connector.

A.4.1.1 Example: Calling a Business Function with Audit Parameters

The following code sample is an example of a dynamic Java connector calling a business function that performs a purchase order, and it shows the parameters required for passing audit information. Step 4 contains the input values for a purchase order business function and step 5 contains the audit data set for the purchase order.

```
//Step 1: Login:
int sessionID = Connector.getInstance().login("user", "pwd", "env", "role");

//Pre-condition: Create the SpecDictionary or BSFNSpecSource
BSFNSpecSource specSource = new OneworldBSFNSpecSource(sessionID);

//The dynamic Java connector enables you to use hash tables to enter parameter⇒
 values. This example code illustrates how to use the HashMap class to enter⇒
 parameter values:

Map input = new HashMap();

//The dynamic Java connector enables you to use hash tables to retrieve output⇒
 values. This example code illustrates how to use the Hash table class to
retrieve⇒
 output values:

Map output = addressbook.getValues();

// Step 2: Look up the business function method from SpecDictionary or BSFNSpec⇒
Source:
BSFNMethod bsfnMethod = (BSFNMethod)specSource.getBSFNMethod("F4311EndDoc");

// Step 3: Create the executable method from the business function metadata:
ExecutableMethod purchaseorder = bsfnMethod.createExecutable();

// Step 4: Set parameter values:
```

```
purchaseorder.setValues(inputParams);
purchaseorder.setValue("mnJobNumber","");
purchaseorder.setValue("mnOrderNumberAssigned","");
purchaseorder.setValue("szRelatedOrderType","");
purchaseorder.setValue("szComputerID","");
purchaseorder.setValue("cUseWorkFiles","");
purchaseorder.setValue("mnProcessID","");
purchaseorder.setValue("mnTransactionID","");

//Step 5: Set GUID,ApplicationID,WorkstationName,IPAddress for Audit

purchaseorder.setGUID("10.139.193.107.3434235324223423");
purchaseorder.setApplicationID("INTEROP")
purchaseorder.setWorkstationName("DENOSCL244");
purchaseorder.setIPAddress("10.139.193.107");

// Step 6: Execute the business function:
try {

BSFNExecutionWarning warning=purchaseorder.execute(sessionID);
```

A.4.2 Reusing Audit Data for Multiple Business Functions

When a third party application runs multiple business functions within one transaction, you can reuse the same audit data that was passed in the first business function call for all subsequent business function calls within the same transaction.

The sample code in this section shows how the salesorder business function is programmed to use the same GUID as the purchaseorder business function. In this example, if you do not configure the salesorder business function to use the same audit values, the API generates default audit information, which includes a new GUID, the word "INTEROP" for the application ID, the local workstation name, and the IP address of the local workstation.

In this example, to configure the other audit parameters, replace GUID with the application ID, workstation name, or IP address.

```
Step 1:

ExecutableMethod purchaseorder = bsfnMethod.createExecutable();
... .

ExecutableMethod saleseorder = bsfnMethod.createExecutable();

...

BSFNExecutionWarning warning=purchaseorder.execute(sessionID);

Step 2:

String firstguid = purchaseorder.getGUID();

Step 3:

salesorder.setGUID(firstguid);

BSFNExecutionWarning warning=purchaseorder.execute(sessionID);
```

A.4.3 Reference Implementation Resources

The files that you need to configure and test the Java Connector reference implementation for auditing are available in a zip file (AuditPurchaseOrderSample.zip) that you can download from the Update Center on the My Oracle Support Web site.

The zip file also contains a ReadMe file that provides additional instructions on how to configure and run this reference implementation.

A.4.4 Running the Java Connector Reference Implementation to Test Auditing

This section provides a sample program that runs the purchaseorder business function with audit parameters. The steps in this section use the Java Connector reference implementation to test the passing of audit information from a Java Connector interoperability transaction.

To run the Java Connector reference implementation to test auditing:

1. Create a Java file with the name ExecuteBSFN_RI.java and paste the code from the sample program below.
2. Update the code with valid JD Edwards EnterpriseOne credentials for login.
3. Update the code with the correct values for the purchaseorder business function parameters.
4. Add the dependent jars for imported classes.
5. Compile the Java file and run it.

Example: Sample program for testing auditing for a Java connector transaction:

```
package javaconn.ri.sample;

import java.util.HashMap;
import java.util.Map;

import com.jdedwards.system.connector.dynamic.callmethod.BSFNExecutionWarning;
import com.jdedwards.system.connector.dynamic.callmethod.ExecutableMethod;
import
com.jdedwards.system.connector.dynamic.callmethod.RequiredParameterNotFound⇒
Exception;
import com.jdedwards.system.connector.dynamic.spec.source.BSFNSpecSource;
import com.jdedwards.system.connector.dynamic.spec.source.OneworldBSFNSpecSource;
import com.jdedwards.system.connector.dynamic.Connector;
import com.jdedwards.system.connector.dynamic.spec.source.*;
import com.jdedwards.system.connector.dynamic.spec.SpecFailureException;
import com.jdedwards.system.connector.dynamic.ServerFailureException;
import com.jdedwards.system.connector.dynamic.SystemException;
import com.jdedwards.system.connector.dynamic.ApplicationException;

public class ExecuteBSFN_RI {
int sessionID;
BSFNSpecSource specSource = null;
private ExecutableMethod poeEndDoc;
private ExecutableMethod poeBeginDoc;
ExecutableMethod poeEditDoc;
public ExecuteBSFN_RI() {
super();
// TODO Auto-generated constructor stub
```

```
}
public void initialize(){
try {
sessionID = Connector.getInstance().login("JDE", "JDE", "JDV812","*ALL");
specSource = new OneworldBSFNSpecSource(sessionID);
} catch (ServerFailureException e) {

e.printStackTrace();
} catch (SpecFailureException e) {

e.printStackTrace();
}
}
public boolean executeBiginDoc() throws SpecFailureException,ServerFailure⇒
Exception
{
Map inputParams = new HashMap();
        Map outputParams = new HashMap();

        inputParams.put("szOrderType", "OP");

        String szBusinessUnit = "            ";
        szBusinessUnit = szBusinessUnit.substring(0, 12 - "M30".length()) + "M30";

        BSFNMethod method = specSource.getBSFNMethod("F4311FSBeginDoc");
        poeBeginDoc = method.createExecutable();

        poeBeginDoc.setValues(inputParams);

        poeBeginDoc.setValue("szBranchPlant",szBusinessUnit);
        poeBeginDoc.setValue("jdOrderDate", "03/04/2007");
        poeBeginDoc.setValue("szUserID", "JDE");
        poeBeginDoc.setValue("mnSupplierNumber", "4242");
        poeBeginDoc.setValue("cHeaderActionCode", "A");
        poeBeginDoc.setValue("cProcessEdits", "1");
        poeBeginDoc.setValue("cUpdateOrWriteToWorkFile", "2");
        poeBeginDoc.setValue("szProgramID", "CORBA");


        poeBeginDoc.setValue("szPurchaseOrderPrOptVersion", "ZJDE0001");
        poeBeginDoc.setValue("szComputerID","DTP-DDHOLAKI-WF");

        try {
        BSFNExecutionWarning warning = poeBeginDoc.execute(sessionID);
            if (warning!=null) {
                System.out.println(warning.toString());
            } else {
                System.out.println("BeginDoc execution Finished");
            }
        }
        catch (ApplicationException e) {
            e.printStackTrace();
            if (e instanceof RequiredParameterNotFoundException){
                String paramName = ((RequiredParameterNotFoundException)e).get⇒
ParamName();
                System.out.println("showError(paramName)");;
            }
            return false;
        } catch (SystemException e) {
            e.printStackTrace();
```

```
                System.exit(1);
            }

            return true;

    }
    public void executeEditDoc() throws SpecFailureException,ServerFailureException
    {
    Map inputParams = new HashMap();
            Map outputParams = new HashMap();

            inputParams.put("szUnformattedItemNumber", "1001");
            inputParams.put("mnQuantityOrdered", "976");
            inputParams.put("mnUnitPrice", "0.0");
            inputParams.put("szLineType",null);
            inputParams.put("szTransactionUoM",null);

            BSFNMethod method = specSource.getBSFNMethod("F4311EditLine");
            poeEditDoc = method.createExecutable();

            poeEditDoc.setValues(inputParams);

            poeEditDoc.setValue("mnJobNumber", poeBeginDoc.getValue("mnJobNumber"));
            System.out.println("done");
            poeEditDoc.setValue("szBranchPlant", poeBeginDoc.getValue("szBranch⇒
    Plant"));
            poeEditDoc.setValue("szComputerID", poeBeginDoc.getValue("szComputerID"));
            poeEditDoc.setValue("cUpdateOrWriteWorkFile","2");
            poeEditDoc.setValue("szOrderType", poeBeginDoc.getValue("szOrderType"));
            poeEditDoc.setValue("cDetailActionCode", "1");
            poeEditDoc.setValue("mnProcessID", poeBeginDoc.getValue("mnProcessID"));
            poeEditDoc.setValue("mnTransactionID",
    poeBeginDoc.getValue("mnTransaction⇒
    ID"));
            poeEditDoc.setValue("mnSupplierNumber", poeBeginDoc.getValue("mnSupplier⇒
    Number"));


            try {
            BSFNExecutionWarning warning = poeEditDoc.execute(sessionID);
                if (warning!=null) {
                    System.out.println(warning.toString());
                }
            } catch (ApplicationException e) {
                e.printStackTrace();
            } catch (SystemException e) {
                e.printStackTrace();
                System.exit(1);
            }

    }
    public void executeEndDoc() throws SpecFailureException,ServerFailureException
    {

    BSFNMethod method = specSource.getBSFNMethod("F4311EndDoc");
    String methodName = method.getName();
    System.out.println("Method name is "+methodName);
    BSFNParameter[] paraList = method.getParameters();
    for (int i=0; i<paraList.length;i++)
    {
```

```
BSFNParameter para = paraList[i];
String name=para.getName();
System.out.println("Name is "+name);
}

// Step 3: Create the executable method from the business function metadata:
ExecutableMethod poeEndDoc = method.createExecutable();
try
{
//    Set GUID,ApplicationID,WorkstationName and IPAddress
poeEndDoc.setGUID("3344556677");
poeEndDoc.setApplicationID("TestRI");
poeEndDoc.setWorkstationName("test-workstation");
poeEndDoc.setIPAddress("10.10.10.10");

poeEndDoc.setValue("mnJobNumber",poeBeginDoc.getValue("mnJobNumber"));
        poeEndDoc.setValue("mnOrderNumberAssigned",poeBeginDoc.getValue("mnOrder⇒
Number"));
        poeEndDoc.setValue("szRelatedOrderType",poeBeginDoc.getValue("szOrder⇒
Type"));
        poeEndDoc.setValue("szComputerID",poeBeginDoc.getValue("szComputerID"));
        poeEndDoc.setValue("cUseWorkFiles","2");
        poeEndDoc.setValue("mnProcessID",poeBeginDoc.getValue("mnProcessID"));
        poeEndDoc.setValue("mnTransactionID",poeBeginDoc.getValue("mnTransaction⇒
ID"));

// Step 5: Execute the business function:

try {

BSFNExecutionWarning warning = poeEndDoc.execute(sessionID);

            if (warning!=null) {
                System.out.println(warning.toString());
            }
} catch (ApplicationException e) {
e.printStackTrace();
} catch (SystemException e) {
    e.printStackTrace();
    System.exit(1);
}
// Step 6: Get return parameter values:
System.out.println("GUID: "+poeEndDoc.getGUID());
System.out.println("Application Name: "+poeEndDoc.getApplicationID());
System.out.println("WorkstationName: "+poeEndDoc.getWorkstationName());
System.out.println("IP Address: "+poeEndDoc.getIPAddress());
System.out.println("mnOrderNumberAssigned "+poeEndDoc.getValueString("mnOrder⇒
NumberAssigned"));
}
finally
{
//Log off and shut down connector if necessary:
Connector.getInstance().logoff(sessionID);
Connector.getInstance().shutDown();
}
}
/**
 * @param args
 */
public static void main(String[] args) {
```

```
ExecuteBSFN_RI ri = new ExecuteBSFN_RI();
try {
ri.initialize();
ri.executeBiginDoc();
ri.executeEditDoc();
ri.executeEndDoc();
} catch (SpecFailureException e) {
e.printStackTrace();
} catch (ServerFailureException e) {
e.printStackTrace();
}

}
}
```

A.5 Configuring a COM Connector to Pass Audit Information

This section provides an example of a generated COM function wrapper for auditing and reference implementation resources.

A.5.1 Example of a Generated COM Function Wrapper with Audit Parameters

The following sample code is an example of a generated COM function wrapper in Visual Basic. This example includes parameters for passing the GUID, application ID, workstation name, and IP address audit values. It creates business and connector objects.

Refer to the SalesOrderEntry sample included with the COM interoperability software for a complete working example of this functionality.

This sample code contains the connCFR object created from the IConnector3 interface, which is used to call audit APIs:

```
' Connection into the OneWorld Environment
Dim jdeConnector As New JDECOMCONNECTOR2Lib.Connector
'Connector interface with roles
Dim connRole As JDECOMCONNECTOR2Lib.IConnector2
'Connector interface for CFR
Dim connCFR As JDECOMCONNECTOR2Lib.IConnector3
' Object needed for error callbacks
Dim WithEvents soeOWInterface As JDECOMCONNECTOR2Lib.OneWorldInterface
' Sales Order Entry Business Object
Dim soeObject As SALESORDERENTRyLib.JDESalesOrderEntry
' Parametersets for soe MBF calls
Dim soeBeginDoc As SALESORDERENTRyLib.D4200310H
Dim soeEndDoc As SALESORDERENTRyLib.D4200310G
Dim soeEditLine As SALESORDERENTRyLib.D4200310F
Dim soeClearWF As SALESORDERENTRyLib.D4200310I

Dim lngAccessNumber As Integer

Dim GUID As String
Dim appID As String
Dim compNam As String
Dim ipAddress As String

Private Sub frmSalesOrder_Load
```

```
Set connRole = jdeConnector
Set connCFR = jdeConnector
lngAccessNumber = connRole.Login("JDE", "JDE", "JDV812", "*ALL")

soeObject = jdeConnector.CreateBusinessObject("SalesOrderEntry.JDESalesOrder⇒
Entry", lngAccessNumber)

soeOWInterface = jdeConnector.CreateBusinessObject("OneWorld.FunctionHelper.1",⇒
 lngAccessNumber)

connCFR.setGUID("000-aaaa-cccc")
connCFR.setAppID("P4210")
connCFR.setWorkStationName("Comp")
connCFR.setIPAddress("1.1.1.1")

soeObject.F4211FSEndDoc(soeEndDoc, soeOWInterface, jdeConnector, lngAccessNumber)

GUID = connCFR.getGUID()
appID = connCFR.getAppID()
compNam = connCFR.getWorkStationName()
ipAddress = connCFR.getIPAddress()

End Sub
```

Note: You must explicitly set the GUID, application ID, workstation name and IP address to blank or a new value in the sample application for every sales order created. If you do not set these values, the system uses the values of the first sales order for all subsequent sales orders created. This happens if a single instance of a connector is used to create multiple sales orders.

The following sample code shows how to generate a new GUID for every business function call. If you pass NULL for the GUID, then the system generates a new GUID:

```
connCFR.setGUID("")
connCFR.setAppID("P4210")
connCFR.setWorkStationName("Comp")
connCFR.setIPAddress("1.1.1.1")

soeObject.F4211FSEndDoc(soeEndDoc, soeOWInterface, jdeConnector, lngAccessNumber)
```

A.5.2 Reference Implementation Resources

The files and resources for configuring and testing a COM connector reference implementation are located in the Update Center on the My Oracle Support Web site.

A.6 Configuring WSG or XPI to Pass Audit Information

An adapter service for a JD Edwards EnterpriseOne business function or database operation contains an audit data section. This section contains these fields for passing audit information to an audit table:

- GUID
- Application ID

- Workstation Name
- IP Address

If audit information is not passed at runtime through the adapter service, then the system generates the audit data.

A.6.1 Adapter Services for EnterpriseOne Business Functions: Auditing Configuration

The business function template in the EnterpriseOne Adapter is embedded with an Audit Data section to pass audit information when you are running the adapter service. Therefore, no special configuration is needed to implement auditing.

A.6.1.1 Multiple Business Functions Within a Single Transaction in a Flow

In a typical scenario, if a flow contains multiple business function adapter services, the audit data is shared across all business function calls within the flow. If custom audit data needs to be passed, include the audit data—which includes a GUID, application ID, workstation name, and IP address—in the flow input, and map the values to the audit data fields in the Audit Data section of the business function adapter in the flow.

The audit fields in the flow service set the audit information for the business function adapter service. If the flow contains multiple adapter services, then all the services will share the single audit information that is provided as the flow input. If multiple adapter services are in a flow, you do not need to map the audit fields in the flow service input document to the Audit Data section of every adapter service. Mapping the audit fields to the first adapter service passes the data to the other adapter services in that flow.

A.6.2 Adapter Services for EnterpriseOne Database Operations: Auditing Configuration

Adapter services for database operations are developed as standalone services that will not be part of any flow. Therefore, there is no need to consider sharing auditing data for multiple adapter services, unlike business function adapter services. The database select service template does not contain an Audit Data section because data is not manipulated for a database select operation.

A.6.3 Reference Implementation Resources

Oracle provides sample files, or reference implementations, that you can use to configure and test an auditing configuration for a WSG or XPI adapter service.

The reference implementations are delivered in this package: PSFT_E1_Adapter_GUID_Samples.zip.

This zip file is available in the Update Center on the My Oracle Support Web site. You must install this package to work with the samples.

B Troubleshooting

This appendix contains the following topics:

- Section B.1, "Audit Processing Error"
- Section B.2, "SQL7032 Problems"

This section contains solutions to some issues that you might encounter using the auditing and signature applications.

B.1 Audit Processing Error

This error is issued if the creation of the Unique ID fails, or whenever a JDB call fails. An Audit Process Error is written in the jde.log.

Error Occurs	Result
Right after a user presses OK or Delete when setting up the audit info.	The user is brought back to the point before clicking the button.
Before a modal form interconnect when clearing the audit info of the parent form.	The interconnect is not processed.
After returning from a modal form interconnect when restoring the audit info of the parent form.	No more ERs are executed on that event. The user is brought back to the point before clicking the button.
Before deleting a row during the OK process on a transaction form (when restoring the audit info for the row to be deleted).	The row is not deleted.
At the end of the delete process during the OK process on a transaction form (when restoring the audit info for the OK process).	The rest of the OK process is not processed, and all transactions are rolled back.

An error can also occur while processing an asynchronous business function, right before calling jdeCallObject(), when setting the audit information for the current business function. As a result, the business function does not run. Instead of displaying a message box, the system writes the following message to the jde.log:

Audit Process Error: There has been an error running the following Business Function: BSFN Name.

B.2 SQL7032 Problems

If you are experiencing SQL7032 problems on the iSeries, and you are running on V5R1, apply the IBM PTF SI04448.

Glossary

Accessor Methods/Assessors

Java methods to "get" and "set" the elements of a value object or other source file.

activity rule

The criteria by which an object progresses from one given point to the next in a flow.

add mode

A condition of a form that enables users to input data.

Advanced Planning Agent (APAg)

A JD Edwards EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of rational databases, flat file format, and other data or message encoding, such as XML.

application server

Software that provides the business logic for an application program in a distributed environment. The servers can be Oracle Application Server (OAS) or WebSphere Application Server (WAS).

Auto Commit Transaction

A database connection through which all database operations are immediately written to the database.

batch processing

A process of transferring records from a third-party system to JD Edwards EnterpriseOne.

In JD Edwards EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than JD Edwards EnterpriseOne to JD Edwards EnterpriseOne Accounts Receivable and JD Edwards EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to JD Edwards EnterpriseOne.

batch server

A server that is designated for running batch processing requests. A batch server typically does not contain a database nor does it run interactive applications.

batch-of-one

A transaction method that enables a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks.

best practices

Non-mandatory guidelines that help the developer make better design decisions.

BPEL

Abbreviation for Business Process Execution Language, a standard web services orchestration language, which enables you to assemble discrete services into an end-to-end process flow.

BPEL PM

Abbreviation for Business Process Execution Language Process Manager, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes.

Build Configuration File

Configurable settings in a text file that are used by a build program to generate ANT scripts. ANT is a software tool used for automating build processes. These scripts build published business services.

build engineer

An actor that is responsible for building, mastering, and packaging artifacts. Some build engineers are responsible for building application artifacts, and some are responsible for building foundation artifacts.

Build Program

A WIN32 executable that reads build configuration files and generates an ANT script for building published business services.

business analyst

An actor that determines if and why an EnterpriseOne business service needs to be developed.

business function

A named set of user-created, reusable business rules and logs that can be called through event rules. Business functions can run a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the application programming interfaces (APIs) that enable them to be called from a form, a database trigger, or a non-JD Edwards EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule

See named event rule (NER).

business service

EnterpriseOne business logic written in Java. A business service is a collection of one or more artifacts. Unless specified otherwise, a business service implies both a published business service and business service.

business service artifacts

Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process.

business service class method

A method that accesses resources provided by the business service framework.

business service configuration files

Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties.

business service cross reference

A key and value data pair used during orchestration. Collectively refers to both the code and the key cross reference in the WSG/XPI based system.

business service cross-reference utilities

Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data.

business service development environment

A framework needed by an integration developer to develop and manage business services.

business services development tool

Otherwise known as JDeveloper.

business service EnterpriseOne object

A collection of artifacts managed by EnterpriseOne LCM tools. Named and represented within EnterpriseOne LCM similarly to other EnterpriseOne objects like tables, views, forms, and so on.

business service framework

Parts of the business service foundation that are specifically for supporting business service development.

business service payload

An object that is passed between an enterprise server and a business services server. The business service payload contains the input to the business service when passed to the business services server. The business service payload contains the results from the business service when passed to the Enterprise Server. In the case of notifications, the return business service payload contains the acknowledgement.

business service property

Key value data pairs used to control the behavior or functionality of business services.

Business Service Property Admin Tool

An EnterpriseOne application for developers and administrators to manage business service property records.

business service property business service group

A classification for business service property at the business service level. This is generally a business service name. A business service level contains one or more business service property groups. Each business service property group may contain zero or more business service property records.

business service property key

A unique name that identifies the business service property globally in the system.

business service property utilities

A utility API used in business service development to access EnterpriseOne business service property data.

business service property value

A value for a business service property.

business service repository

A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network.

business services server

The physical machine where the business services are located. Business services are run on an application server instance.

business services source file or business service class

One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler.

business service value object template

The structural representation of a business service value object used in a C-business function.

Business Service Value Object Template Utility

A utility used to create a business service value object template from a business service value object.

business services server artifact

The object to be deployed to the business services server.

business view

A means for selecting specific columns from one or more JD Edwards EnterpriseOne application tables whose data is used in an application or report. A business view does not select specific rows, nor does it contain any actual data. It is strictly a view through which you can manipulate data.

central objects merge

A process that blends a customer's modifications to the objects in a current release with objects in a new release.

central server

A server that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers. In a typical JD Edwards EnterpriseOne installation, the software is loaded on to one machine—the central server. Then, copies of the software are pushed out or downloaded to various workstations attached to it. That way, if the software is altered or corrupted through its use on workstations, an original set of objects (central objects) is always available on the central server.

charts

Tables of information in JD Edwards EnterpriseOne that appear on forms in the software.

check-in repository

A repository for developers to check in and check out business service artifacts. There are multiple check-in repositories. Each can be used for a different purpose (for example, development, production, testing, and so on).

checksum

A fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. JD Edwards EnterpriseOne uses the checksum to verify the integrity of packages that have been downloaded by recomputing the checksum of the downloaded package and comparing it with the checksum of the original package. The procedure that yields the checksum from the data is called a checksum function or checksum algorithm. JD Edwards EnterpriseOne uses the MD5 and STA-1 checksum algorithms.

connector

Component-based interoperability model that enables third-party applications and JD Edwards EnterpriseOne to share logic and data. The JD Edwards EnterpriseOne connector architecture includes Java and COM connectors.

Control Table Workbench

An application that, during the Installation Workbench processing, runs the batch applications for the planned merges that update the data dictionary, user-defined codes, menus, and user override tables.

control tables merge

A process that blends a customer's modifications to the control tables with the data that accompanies a new release.

correlation data

The data used to tie HTTP responses with requests that consist of business service name and method.

credentials

A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token.

cross-reference utility services

Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data.

database credentials

A valid database username/password.

database server

A server in a local area network that maintains a database and performs searches for client computers.

Data Source Workbench

An application that, during the Installation Workbench process, copies all data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the system-release number data source. It also updates the Data Source Plan detail record to reflect completion.

deployment artifacts

Artifacts that are needed for the deployment process, such as servers, ports, and such.

deployment server

A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.

direct connect

A transaction method in which a client application communicates interactively and directly with a server application.

See also batch-of-one and store-and-forward.

Do Not Translate (DNT)

A type of data source that must exist on the iSeries because of BLOB restrictions.

embedded application server instance

An OC4J instance started by and running wholly within JDeveloper.

edit code

A code that indicates how a specific value for a report or a form should appear or be formatted. The default edit codes that pertain to reporting require particular attention because they account for a substantial amount of information.

edit mode

A condition of a form that enables users to change data.

edit rule

A method used for formatting and validating user entries against a predefined rule or set of rules.

Electronic Data Interchange (EDI)

An interoperability model that enables paperless computer-to-computer exchange of business transactions between JD Edwards EnterpriseOne and third-party systems. Companies that use EDI must have translator software to convert data from the EDI standard format to the formats of their computer systems.

embedded event rule

An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with the business function event rule.

Employee Work Center

A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages.

enterprise server

A server that contains the database and the logic for JD Edwards EnterpriseOne.

Enterprise Service Bus (ESB)

Middleware infrastructure products or technologies based on web services standards that enable a service-oriented architecture using an event-driven and XML-based messaging framework (the bus).

EnterpriseOne administrator

An actor responsible for the EnterpriseOne administration system.

EnterpriseOne credentials

A user ID, password, environment, and role used to validate a user of EnterpriseOne.

EnterpriseOne development client

Historically called “fat client,” a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools.

EnterpriseOne extension

A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard

is a specific example of an extension.

EnterpriseOne object

A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects.

EnterpriseOne process

A software process that enables JD Edwards EnterpriseOne clients and servers to handle processing requests and run transactions. A client runs one process, and servers can have multiple instances of a process. JD Edwards EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don't have to wait if the server is particularly busy.

EnterpriseOne resource

Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users.

Environment Workbench

An application that, during the Installation Workbench process, copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the system-release number data source. It also updates the Environment Plan detail record to reflect completion.

escalation monitor

A batch process that monitors pending requests or activities and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time.

event rule

A logic statement that instructs the system to perform one or more operations based on an activity that can occur in a specific application, such as entering a form or exiting a field.

explicit transaction

Transaction used by a business service developer to explicitly control the type (auto or manual) and the scope of transaction boundaries within a business service.

exposed method or value object

Published business service source files or parts of published business service source files that are part of the published interface. These are part of the contract with the customer.

fast path

A command prompt that enables the user to move quickly among menus and applications by using specific commands.

file server

A server that stores files to be accessed by other computers on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files, but also manages them and maintains order as network users request files and make changes to these files.

final mode

The report processing mode of a processing mode of a program that updates or creates data records.

foundation

A framework that must be accessible for execution of business services at runtime. This includes, but is not limited to, the Java Connector and JDBj.

FTP server

A server that responds to requests for files via file transfer protocol.

HTTP Adapter

A generic set of services that are used to do the basic HTTP operations, such as GET, POST, PUT, DELETE, TRACE, HEAD, and OPTIONS with the provided URL.

instantiate

A Java term meaning “to create.” When a class is instantiated, a new instance

is created.

integration developer

The user of the system who develops, runs, and debugs the EnterpriseOne business services. The integration developer uses the EnterpriseOne business services to develop these components.

integration point (IP)

The business logic in previous implementations of EnterpriseOne that exposes a document level interface. This type of logic used to be called XBPs. In EnterpriseOne 8.11, IPs are implemented in Web Services Gateway powered by webMethods.

integration server

A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems.

integrity test

A process used to supplement a company’s internal balancing procedures by locating and reporting balancing problems and data inconsistencies.

interface table

See Z table.

internal method or value object

Business service source files or parts of business service source files that are not part of the published interface. These could be private or protected methods. These could be value objects not used in published methods.

interoperability model

A method for third-party systems to connect to or access JD Edwards EnterpriseOne.

in-your-face error

In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.

jargon

An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object.

Java application server

A component-based server that resides in the middle-tier of a server-centric architecture. This server provides middleware services for security and state maintenance, along with data access and persistence.

JDBNET

A database driver that enables heterogeneous servers to access each other's data.

JDEBASE Database Middleware

A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access.

JDECallObject

An API used by business functions to invoke other business functions.

jde.ini

A JD Edwards EnterpriseOne file (or member for iSeries) that provides the runtime settings required for JD Edwards EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running JD Edwards EnterpriseOne. This includes workstations and servers.

JDEIPC

Communications programming tools used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes.

jde.log

The main diagnostic log file of JD Edwards EnterpriseOne. This file is always located in the root directory on the primary drive and contains status and error messages from the startup and operation of JD Edwards EnterpriseOne.

JDENET

A JD Edwards EnterpriseOne proprietary communications middleware package. This package is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all JD Edwards EnterpriseOne supported platforms.

JDeveloper Project

An artifact that JDeveloper uses to categorize and compile source files.

JDeveloper Workspace

An artifact that JDeveloper uses to organize project files. It contains one or more project files.

JMS Queue

A Java Messaging service queue used for point-to-point messaging.

listener service

A listener that listens for XML messages over HTTP.

local repository

A developer's local development environment that is used to store business service artifacts.

Location Workbench

An application that, during the Installation Workbench process, copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the system data source.

logic server

A server in a distributed network that provides the business logic for an application program. In a typical configuration, pristine objects are replicated on to the logic server from the central server. The logic server, in conjunction with workstations, actually performs the processing required when JD Edwards EnterpriseOne software runs.

MailMerge Workbench

An application that merges Microsoft Word 6.0 (or higher) word-processing documents with JD Edwards EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment.

Manual Commit transaction

A database connection where all database operations delay writing to the database until a call to commit is made.

master business function (MBF)

An interactive master file that serves as a central location for adding, changing, and updating information in a database. Master business functions pass information between data entry forms and the appropriate tables. These master functions provide a common set of functions that contain all of the necessary default and editing rules for related programs. MBFs contain logic that ensures the integrity of adding, updating, and deleting information from databases.

master table

See published table.

media storage object

Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.

message center

A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user.

messaging adapter

An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues.

messaging server

A server that handles messages that are sent for use by other programs using a messaging API. Messaging servers typically employ a middleware program to perform their functions.

Monitoring Application

An EnterpriseOne tool provided for an administrator to get statistical information for various EnterpriseOne servers, reset statistics, and set notifications.

named event rule (NER)

Encapsulated, reusable business logic created using event rules, rather that C programming. NERs are also called business function event rules. NERs can be reused in multiple places by multiple programs. This modularity lends itself to streamlining, reusability of code, and less work.

Object Configuration Manager (OCM)

In JD Edwards EnterpriseOne, the object request broker and control center for the runtime environment. OCM keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, OCM directs access to it using defaults and overrides for a given environment and user.

Object Librarian

A repository of all versions, applications, and business functions reusable in building applications. Object Librarian provides check-out and check-incapabilities for developers, and it controls the creation, modification, and use of JD Edwards EnterpriseOne objects. Object Librarian supports multiple environments (such as production and development) and enables objects to be easily moved from one environment to another.

Object Librarian merge

A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release.

Open Data Access (ODA)

An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation.

Output Stream Access (OSA)

An interoperability model that enables you to set up an interface for JD Edwards EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.

package

JD Edwards EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the installation program can find them. It is point-in-time snapshot of the central objects on the deployment server.

package build

A software application that facilitates the deployment of software changes and new applications to existing users. Additionally, in JD Edwards EnterpriseOne, a package build can be a compiled version of the software. When you upgrade your version of the ERP software, for example, you are said to take a package build.

Consider the following context: "Also, do not transfer business functions into the production path code until you are ready to deploy, because a global build of business functions done during a package build will automatically include the new functions." The process of creating a package build is often referred to, as it is in this example, simply as "a package build."

package location

The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\package name. The subdirectories under this path are where the replicated objects for the package are placed. This is also referred to as where the package is built or stored.

Package Workbench

An application that, during the Installation Workbench process, transfers the package information tables from the Planner data source to the system-release number data source. It also updates the Package Plan detail record to reflect completion.

Pathcode Directory

The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored.

patterns

General repeatable solutions to a commonly occurring problem in software design. For business service development, the focus is on the object relationships and interactions.

For orchestrations, the focus is on the integration patterns (for example, synchronous and asynchronous request/response, publish, notify, and receive/reply).

print server

The interface between a printer and a network that enables network clients to connect to the printer and send their print jobs to it. A print server can be a computer, separate hardware device, or even hardware that resides inside of the printer itself.

pristine environment

A JD Edwards EnterpriseOne environment used to test unaltered objects with JD Edwards EnterpriseOne demonstration data or for training classes. You must have this environment so that you can compare pristine objects that you modify.

processing option

A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed, to specify date ranges, to supply runtime values that regulate program execution, and so on.

production environment

A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software.

Production Published Business Services Web Service

Published business services web service deployed to a production application server.

program temporary fix (PTF)

A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks.

project

In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.

promotion path

The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):

11>21>26>28>38>01

In this path, 11 equals new project pending review, 21 equals programming, 26 equals QA test/review, 28 equals QA test/review complete, 38 equals in production, 01 equals complete. During the normal project promotion cycle, developers check objects out of and into the development path code and then promote them to the prototype path code. The objects are then moved to the productions path code before declaring them complete.

proxy server

A server that acts as a barrier between a workstation and the internet so that the enterprise can ensure security, administrative control, and caching service.

published business service

EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems.

published business service identification information

Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL.

published business service web service

Published business services components packaged as J2EE Web Service (namely, a J2EE EAR file that contains business service classes, business service foundation, configuration files, and web service artifacts).

published table

Also called a master table, this is the central copy to be replicated to other machines. Residing on the publisher machine, the F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

publisher

The server that is responsible for the published table. The F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

QBE

An abbreviation for query by example. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.

real-time event

A message triggered from EnterpriseOne application logic that is intended for external systems to consume.

refresh

A function used to modify JD Edwards EnterpriseOne software, or subset of it, such as a table or business data, so that it functions at a new release or cumulative update level.

replication server

A server that is responsible for replicating central objects to client machines.

rules

Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards.

secure by default

A security model that assumes that a user does not have permission to execute an object unless there is a specific record indicating such permissions.

Secure Socket Layer (SSL)

A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

selection

Found on JD Edwards EnterpriseOne menus, a selection represents functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

serialize

The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed.

Server Workbench

An application that, during the Installation Workbench process, copies the server configuration files from the Planner data source to the system-release number data source. The application also updates the Server Plan detail record to reflect completion.

SOA

Abbreviation for Service Oriented Architecture.

softcoding

A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process.

source repository

A repository for HTTP adapter and listener service development environment artifacts.

Specification merge

A merge that comprises three merges: Object Librarian merge, Versions List merge, and Central Objects merge. The merges blend customer modifications with data that accompanies a new release.

specification

A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.

Specification Table Merge Workbench

An application that, during the Installation Workbench process, runs the batch applications that update the specification tables.

SSL Certificate

A special message signed by a certificate authority that contains the name of a user and that user's public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in the user's public key.

store-and-forward

The mode of processing that enables users who are disconnected from a server to enter transactions and then later connect to the server to upload those transactions.

subscriber table

Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table.

super class

An inheritance concept of the Java language where a class is an instance of something, but is also more specific. “Tree” might be the super class of “Oak” and “Elm,” for example.

table access management (TAM)

The JD Edwards EnterpriseOne component that handles the storage and retrieval of use-defined data. TAM stores information, such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions.

Table Conversion Workbench

An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table conversion

An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table event rules

Logic that is attached to database triggers that runs whenever the action specified by the trigger occurs against the table. Although JD Edwards EnterpriseOne enables event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

terminal server

A server that enables terminals, microcomputers, and other devices to connect to a network or host computer or to devices attached to that particular computer.

transaction processing (TP) monitor

A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and may include programs that validate data and format terminal screens.

transaction processing method

A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel).

transaction set

An electronic business transaction (electronic data interchange standard document) made up of segments.

trigger

One of several events specific to data dictionary items. You can attach logic to a data dictionary item that the system processes automatically when the event occurs.

triggering event

A specific workflow event that requires special action or has defined consequences or resulting actions.

user identification information

User ID, role, or *public.

User Overrides merge

Adds new user override records into a customer's user override table.

value object

A specific type of source file that holds input or output data, much like a data structure passes data. Value objects can be exposed (used in a published business service) or internal, and input or output. They are comprised of simple and complex elements and accessories to those elements.

versioning a published business service

Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces.

Versions List merge

The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release, as well as their processing options data.

visual assist

Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control.

vocabulary override

An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report.

web application server

A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions.

web server

A server that sends information as requested by a browser, using the TCP/IP set of protocols. A web server can do more than just coordination of requests from browsers; it can do anything a normal server can do, such as house applications or data. Any computer can be turned into a web server by installing server software and connecting the machine to the internet.

Web Service Description Language (WSDL)

An XML format for describing network services.

Web Service Inspection Language (WSIL)

An XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information should be made.

web service softcoding record

An XML document that contains values that are used to configure a web service proxy. This document identifies the endpoint and conditionally includes security information.

web service softcoding template

An XML document that provides the structure for a soft coded record.

Where clause

The portion of a database operation that specifies which records the database operation will affect.

Windows terminal server

A multiuser server that enables terminals and minimally configured computers to display Windows applications even if they are not capable of running Windows software themselves. All client processing is performed centrally at the Windows terminal server and only display, keystroke, and mouse commands are transmitted over the network to the client terminal device.

wizard

A type of JDeveloper extension used to walk the user through a series of steps.

workbench

A program that enables users to access a group of related programs from a single entry point. Typically, the programs that you access from a workbench are used to complete a large business process. For example, you use the JD Edwards EnterpriseOne Payroll Cycle Workbench (P07210) to access all of the programs that the system uses to process payroll, print payments, create payroll reports, create journal entries, and update payroll history. Examples of JD Edwards EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench.

workflow

The automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.

workgroup server

A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing.

XAPI events

A service that uses system calls to capture JD Edwards EnterpriseOne transactions as they occur and then calls third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested notification when the specified transactions occur to return a response.

XML CallObject

An interoperability capability that enables you to call business functions.

XML Dispatch

An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses.

XML List

An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks.

XML Service

An interoperability capability that enables you to request events from one JD Edwards EnterpriseOne system and receive a response from another JD Edwards EnterpriseOne system.

XML Transaction

An interoperability capability that enables you to use a predefined transaction type to send information to or request information from JD Edwards EnterpriseOne. XML transaction uses interface table functionality.

XML Transaction Service (XTS)

Transforms an XML document that is not in the JD Edwards EnterpriseOne format into an XML document that can be processed by JD Edwards EnterpriseOne. XTS then transforms the response back to the request originator XML format.

Z event

A service that uses interface table functionality to capture JD Edwards EnterpriseOne transactions and provide notification to third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested to be notified when certain transactions occur.

Z table

A working table where non-JD Edwards EnterpriseOne information can be stored and then processed into JD Edwards EnterpriseOne. Z tables also can be used to retrieve JD Edwards EnterpriseOne data. Z tables are also known as interface tables.

Z transaction

Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database.

Index