PeopleSoft®

# PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Printing Administration Technologies

**August 2005**

PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Printing Administration Technologies
SKU E1_TOOLS895TRP-B 0805

# Contents

**General Preface**

**About This PeopleBook Preface ..... vii**

PeopleSoft Application Prerequisites ..... vii

PeopleSoft Application Fundamentals ..... vii

Documentation Updates and Printed Documentation ..... viii

Obtaining Documentation Updates ..... viii

Ordering Printed Documentation ..... viii

Additional Resources ..... ix

Typographical Conventions and Visual Cues ..... x

Typographical Conventions ..... x

Visual Cues ..... xi

Country, Region, and Industry Identifiers ..... xii

Currency Codes ..... xii

Comments and Suggestions ..... xii

Common Elements Used in PeopleBooks ..... xiii

**Preface**

**PeopleSoft EnterpriseOne Tools Development Tools: Report Printing Administration Technologies Preface ..... xv**

Development Tools: Report Printing Administration Technologies Companion Documentation ..... xv

**Chapter 1**

**Getting Started with PeopleSoft EnterpriseOne Tools Development Tools: Report Printing Administration Technologies ..... 1**

Development Tools: Report Printing Administration Technologies Overview ..... 1

Development Tools: Report Printing Administration Technologies Implementation ..... 1

Report Printing Administration Technologies Implementation Steps ..... 1

**Chapter 2**

**Understanding PeopleSoft EnterpriseOne Development Tools: Report Printing Administration Technologies ..... 3**

Report Output ..... 3

Output Management ..... 3

**Chapter 3**
**Defining Print Properties for Reports.....5**
Understanding Print Properties.....5
Modifying Print Properties.....5
Understanding Designated Printers.....5
Understanding Paper Types.....6
Understanding Exporting to CSV.....6
Understanding OSA Interfaces.....8
Understanding the Initialize Logical Printer Name System Function.....8
Prerequisite.....8
Defining Printers in Report Design Aid.....8
Selecting Paper Types in Report Design Aid.....8
Exporting to CSV in Report Design Aid.....9

**Chapter 4**
**Understanding Print Properties at Runtime.....11**
Batch Version Submission.....11
Jobs Submitted From the Microsoft Windows Client.....12
Printer Information.....12
The IFS on the iSeries.....13
Printer Selections at Runtime.....13
Paper Type Selections at Runtime.....13
Print Orientation Selections at Runtime.....14
Export to CSV Option at Runtime.....14
Print Settings in the jde.ini.....14

**Chapter 5**
**Working with Report Printing Administration.....17**
Understanding Report Printing Administration.....17
Working With the Printer Application.....17
Understanding the Printers Application.....18
Understanding Null Pass-Through Print Filters.....20
Forms Used to Add Printers.....21
Adding Printers.....21
Defining Default Printers.....25
Modifying Printers.....27
Copying Printers.....27
Deleting Printers.....27

Deleting Paper Types....27
Adding Null Pass-through Print Filters....27
Searching for Incorrect Printer Records....28
Setting Up Barcode Fonts....29
Understanding Barcode Fonts....29
Forms Used to Set Up Printers to Use Barcode Fonts....29
Setting Up Printers to Use Barcode Fonts....29
Modifying Barcode Printer Information....30
Copying Barcode Printer Information for New Printers....30
Deleting Barcode Support Information From Printers....31
Understanding Multiple Code Sets for PCL....31
Multiple Code Sets for PCL Printing....31
The Order of Precedence for PCL Printing....32
Designing Reports to Print on Line Printers....32
Understanding Reports Designed to Print on Line Printers....32
Understanding Defining Remote iSeries Line Printers to Print Multiple Copies of Reports....33
Prerequisite....33
Modifying Reports to Print on Line Printers....33
Printing Multiple Copies of Reports to Remote iSeries Line Printers....34
Printing Reports....34
Batch Versions at Submission....34
Batch Versions Processed on the Server....35
Batch Versions Processed Locally From the Microsoft Windows Client....35
Print-Time Characteristics....36
Print Settings for Batch Versions....36

**Chapter 6**
**Working with Output Stream Access....39**
Understanding Output Stream Access....39
Creating OSA Libraries....41
OSA Libraries....41
Function Parameters....41
OSA Documents....43
Include Files....43
File Locations and Names....47
OSASample Source Code....48
Creating and Associating OSA Interfaces....67
Understanding OSA Interfaces....67
Forms Used to Create and Associate OSA Interfaces....69

Creating OSA Interface Definitions.....69
Associating an OSA Interface with an Object.....70

**Glossary of PeopleSoft Terms.....73**

**Index .....83**

# About This PeopleBook Preface

PeopleBooks provide you with the information that you need to implement and use PeopleSoft applications.

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

**Note.** PeopleBooks document only page elements, such as fields and check boxes, that require additional explanation. If a page element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft applications are defined in this preface.

## PeopleSoft Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use PeopleSoft applications.

You might also want to complete at least one PeopleSoft introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft menus, and pages, forms, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your PeopleSoft applications most effectively.

## PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft applications.

**Note.** Application fundamentals PeopleBooks are not applicable to the PeopleTools product.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Most PeopleSoft product lines have a version of the application fundamentals PeopleBook. The preface of each PeopleBook identifies the application fundamentals PeopleBooks that are associated with that PeopleBook.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across one or more product lines. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals PeopleBooks. They provide the starting points for fundamental implementation tasks.

# Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

## Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on the PeopleSoft Customer Connection website. Through the Documentation section of PeopleSoft Customer Connection, you can download files to add to your PeopleBook Library. You'll find a variety of useful and timely materials, including updates to the full PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM.

**Important!** Before you upgrade, you must check PeopleSoft Customer Connection for updates to the upgrade instructions. PeopleSoft continually posts updates as the upgrade process is refined.

### See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

## Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

### Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

### Telephone

Contact MMA Partners at 877 588 2525.

### Email

Send email to MMA Partners at peoplebookspress@mmapartner.com.

### See Also

PeopleSoft Customer Connection, https://www.peoplesoft.com/corp/en/login.jsp

## Additional Resources

The following resources are located on the PeopleSoft Customer Connection website:

| Resource | Navigation |
|---|---|
| Application maintenance information | Updates + Fixes |
| Business process diagrams | Support, Documentation, Business Process Maps |
| Interactive Services Repository | Interactive Services Repository |
| Hardware and software requirements | Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Hardware and Software Requirements |
| Installation guides | Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Installation Guides and Notes |
| Integration information | Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Pre-built Integrations for PeopleSoft Enterprise and PeopleSoft EnterpriseOne Applications |
| Minimum technical requirements (MTRs) (EnterpriseOne only) | Implement, Optimize + Upgrade, Implementation Guide, Supported Platforms |
| PeopleBook documentation updates | Support, Documentation, Documentation Updates |
| PeopleSoft support policy | Support, Support Policy |
| Prerelease notes | Support, Documentation, Documentation Updates, Category, Prerelease Notes |
| Product release roadmap | Support, Roadmaps + Schedules |
| Release notes | Support, Documentation, Documentation Updates, Category, Release Notes |

| Resource | Navigation |
|---|---|
| Release value proposition | Support, Documentation, Documentation Updates, Category, Release Value Proposition |
| Statement of direction | Support, Documentation, Documentation Updates, Category, Statement of Direction |
| Troubleshooting information | Support, Troubleshooting |
| Upgrade documentation | Support, Documentation, Upgrade Documentation and Scripts |

# Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

## Typographical Conventions

This table contains the typographical conventions that are used in PeopleBooks:

| Typographical Convention or Visual Cue | Description |
|---|---|
| **Bold** | Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call. |
| *Italics* | Indicates field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply.<br>We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*. |
| KEY+KEY | Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key. |
| `Monospace font` | Indicates a PeopleCode program or other code example. |

| Typographical Convention or Visual Cue | Description |
|---|---|
| “ ” (quotation marks) | Indicate chapter titles in cross-references and words that are used differently from their intended meanings. |
| . . . (ellipses) | Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax. |
| { } (curly braces) | Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe ( \| ). |
| [ ] (square brackets) | Indicate optional items in PeopleCode syntax. |
| & (ampersand) | When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object.<br>Ampersands also precede all PeopleCode variables. |

# Visual Cues

PeopleBooks contain the following visual cues.

## Notes

Notes indicate information that you should pay particular attention to as you work with the PeopleSoft system.

**Note.** Example of a note.

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

**Important!** Example of an important note.

## Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

***Warning!*** Example of a warning.

## Cross-References

PeopleBooks provide cross-references either under the heading “See Also” or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

## Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

### Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

### Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in PeopleBooks:

- Asia Pacific
- Europe
- Latin America
- North America

### Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in PeopleBooks:

- USF (U.S. Federal)
- E&G (Education and Government)

## Currency Codes

Monetary amounts are identified by the ISO currency code.

# Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

# Common Elements Used in PeopleBooks

**Address Book Number** Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant id, participant number, and so on.

**As If Currency Code** Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code allows you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered.

**Batch Number** Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002).

**Batch Date** Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date.

**Batch Status** Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:

*Blank:* Batch is unposted and pending approval.

*A:* The batch is approved for posting, has no errors and is in balance, but it has not yet been posted.

*D:* The batch posted successfully.

*E:* The batch is in error. You must correct the batch before it can post.

*P:* The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to E.

*U:* The batch is temporarily unavailable because someone is working with it, or the batch appears to be in use because a power failure occurred while the batch was open.

**Branch/Plant** Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit.

**Business Unit** Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant.

**Category Code** Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization.

**Company** Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet.

**Currency Code**

Enter the three-character code that represents the currency of the transaction. PeopleSoft EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table.

**Document Company**

Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.

If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.

If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.

**Document Number**

Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program.

**Document Type**

Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. PeopleSoft EnterpriseOne reserves these prefixes for the document types indicated:

*P:* Accounts payable documents.

*R:* Accounts receivable documents.

*T:* Time and pay documents.

*I:* Inventory documents.

*O:* Purchase order documents.

*S:* Sales order documents.

**Effective Date**

Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates:

- The date on which a change of address becomes effective.
- The date on which a lease becomes effective.
- The date on which a price becomes effective.
- The date on which the currency exchange rate becomes effective.
- The date on which a tax rate becomes effective.

**Fiscal Period** and **Fiscal Year**

Enter a number that identifies the general ledger period and year. For many programs, you can leave these fields blank to use the current fiscal period and year defined in the Company Names & Number program (P0010).

**G/L Date** (general ledger date)

Enter the date that identifies the financial period to which a transaction will be posted. The system compares the date that you enter on the transaction to the fiscal date pattern assigned to the company to retrieve the appropriate fiscal period number and year, as well as to perform date validations.

# PeopleSoft EnterpriseOne Tools Development Tools: Report Printing Administration Technologies Preface

This preface discusses Development Tools: Report Printing Administration Technologies companion documentation.

## Development Tools: Report Printing Administration Technologies Companion Documentation

Additional, essential information describing the setup and design of PeopleSoft EnterpriseOne Tools resides in companion documentation. The companion documentation consists of important topics that apply to PeopleSoft EnterpriseOne Report Printing Administration Technologies as well as other PeopleSoft EnterpriseOne Tools. You should be familiar with the contents of these companion PeopleBooks:

- Development Tools: Batch Versions
- Development Tools: Report Design Aid
- Server and Workstation Administration
- System Administration

### See Also

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Batch Versions*, "Getting Started with PeopleSoft EnterpriseOne Tools Development Tools: Batch Versions"

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Design Aid*, "Getting Started with PeopleSoft EnterpriseOne Tools Development Tools: Report Design Aid"

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Server & Workstation Administration*, "Getting Started with PeopleSoft EnterpriseOne Tools Server and Workstation Administration"

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: System Administration*, "Getting Started with PeopleSoft EnterpriseOne Tools System Administration"

# CHAPTER 1

# Getting Started with PeopleSoft EnterpriseOne Tools Development Tools: Report Printing Administration Technologies

This chapter discusses:

- Report Printing Administration Technologies Overview.
- Report Printing Administration Technologies Implementation.

## Development Tools: Report Printing Administration Technologies Overview

Development Tools: Report Printing Administration Technologies addresses the printing properties available in Report Design Aid, the printing properties presented at runtime, how to define printers for reporting, and the different output options available for PeopleSoft EnterpriseOne reports.

## Development Tools: Report Printing Administration Technologies Implementation

This section provides an overview of the steps that are required to implement Development Tools: Report Printing Administration Technologies.

### Report Printing Administration Technologies Implementation Steps

This section provides an overview of the steps that are required to implement Development Tools: Report Printing Administration Technologies.

In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides and troubleshooting information. A complete list of these resources appears in the preface in *About These PeopleBooks* with information about where to find the most current version of each.

This table lists the steps for the Development Tools: Report Printing Administration Technologies implementation.

| Step | Reference |
| --- | --- |
| 1. Set up permissions to access and use Object Management Workbench (OMW) and the Printers application using Security Workbench. | *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Security Administration*, "Using Security Workbench," Managing Application Security |
| 2. Add yourself to the system in a developer role so that you have permissions to create and modify PeopleSoft EnterpriseOne objects. | *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Object Management Workbench*, "Configuring User Roles and Allowed Actions," Setting Up User Roles |
| 3. Set up permissions to create OMW projects. | *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Object Management Workbench*, "Configuring User Roles and Allowed Actions," Setting Up Allowed User Actions |
| 4. Set up save locations to enable you to save PeopleSoft EnterpriseOne objects that are not ready to be checked in. | *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Object Management Workbench*, "Configuring Object Save Locations" |

# CHAPTER 2

# Understanding PeopleSoft EnterpriseOne Development Tools: Report Printing Administration Technologies

This chapter discusses:

- Report output.
- Output management.

## Report Output

In Report Design Aid, you create reports and define specific printing properties to affect the report output. The report developer can process reports from the Microsoft Windows client. The batch engine can process reports on various servers or on the workstation.

After development is complete, the reports and associated batch versions are checked in and advanced through the development cycle. The system administrator then builds a package and deploys the reports and batch versions to the enterprise server. The reports and batch versions are generated to HTML so that they can be run from the web client.

Report output from the Microsoft Windows client can be in the form of viewing the report online, exporting the report to a comma separated values (CSV) format, exporting the report using Output Stream Access (OSA), or sending the report to a printer. After report processing is complete, Output Management handles the generation and output of the report.

From the web client, the batch engine processes reports on various servers. Report output from the web client can be in the form of exporting the report to a CSV format, exporting the report using Output Stream Access, or sending the report to a printer. Once the report has been processed, you can view the report using options on the Row menu. Viewing the report on screen is not an option at runtime from the web client.

## Output Management

Output management refers to managing the different output options available for viewing a report. You can view reports in different file types, send them to different printers, and create output in different forms or paper sizes. PeopleSoft EnterpriseOne accommodates simple output processes such as viewing the PDF of a report online or sending it to a network printer. You can also use more complex processes such as sending versions of a report to different printer drawers or defining versions to print to different printers across the country.

Some output options are defined in initialization files. For the Microsoft Windows client, the jde.ini is read at runtime by the Microsoft Windows client. For the web client, the jas.ini is read at runtime by the JAS server.

# CHAPTER 3

# Defining Print Properties for Reports

This chapter provides an overview of print properties and discusses how to modify print properties.

## Understanding Print Properties

You define print properties in the Printers application when you add a new printer to the system. You define the printer for use by a specific user role or all users. You then associate the printer with a specific report, a specific version of a report, or for all reports.

Report Design Aid also incorporates printing properties to determine the format of the report output. The basic hierarchy of print properties are as follows:

1. Print properties defined at runtime override all other printer definitions: Printers application, report template, and batch version definitions.
2. Print properties defined in batch versions override the properties defined in the associated report template.
3. Print properties defined in Report Design Aid override the properties defined in the Printers application.
4. Print properties defined in the Printers application are the default properties used for batch applications.

**Note.** When you modify print properties in an existing report template, the modifications are not reflected in any of the versions that exist at the time of the modification.

## Modifying Print Properties

This section provides overviews of designated printers, paper types, exporting to CSV (Comma Separated Values), OSA (Output Stream Access) interfaces, and the Initialize Logical Printer Name system function, lists the prerequisite, and discusses how to:

- Define printers in Report Design Aid.
- Select paper types in Report Design Aid.
- Export to CSV in Report Design Aid.

### Understanding Designated Printers

The system administrator defines the default printer to be used with batch processes. The printer is associated with a user role or with *PUBLIC, a default value that includes all users. The printer is associated with a batch application, a batch version, or *ALL (a default value that includes all batch applications or all batch versions for a specific batch application).

A printer associated with a specific user role overrides a printer associated with *PUBLIC and a specific batch process. You can override these printer definitions by selecting a printer from Print Setup in Report Design Aid. The printer you select is stored in the print specifications, causing the report to always print to the printer you defined unless overridden at runtime.

When you submit a batch version to a printer, the system looks first for the printer defined in Report Design Aid, if no printer is defined, the system uses the default printer defined in the Printers application. The system determines a printer based on a hierarchical structure:

1. Printer defined at runtime.
2. Printer defined in Report Design Aid.

   The printer definition becomes part of the report specifications.
3. Printer defined in the Printer application using this hierarchy:
   a. Specific report defined for the user or role.
   b. Specific report defined for *PUBLIC.
   c. All reports defined for the user or role.
   d. All reports defined for *PUBLIC.

## Understanding Paper Types

You can select from predefined paper sizes, or you can enter custom paper dimensions for printing reports.

The standard predefined selections available in the Printers application are A4, Legal, and Letter. You must define one of these selections as the default paper type. You can override this default paper type from Print Setup in Report Design Aid.

The paper types defined in the Printers application are stored in the F986162 table. Report Design Aid inherits the paper size from this table. You must define additional paper types for use in Report Design Aid using the Printers application.

You can define custom paper sizes in the Printers application using a selection of different units of measurement. The minimum definable width in inches is two inches, and the maximum is 21. The minimum definable height in inches is two inches, and the maximum is 24. In Report Design Aid, you can also define custom paper sizes from the Print Setup form. Definitions that you set up in Report Design Aid override the definitions set up in the Printers application.

## Understanding Exporting to CSV

A CSV file has all of the commas stripped from the data fields. In addition to viewing the CSV file, you can manipulate the report data after the report finishes processing. To view report data in a spreadsheet program, such as Excel or Lotus, select the export to a CSV option. You can select the CSV option from multiple locations:

- In Report Design Aid for the report template.

  Use this option to ensure that the report is output to a CSV file every time *any* of the associated batch version are run.

  Select a report template and in Report Design Aid, select the Export to CSV option in Print Setup.

  The Export to CSV option is selected by the system at runtime. If a report template is defined to export to CSV for every instance, you can clear the Export to CSV option at runtime when you do not want the batch version to export to a CSV file for a single submission.

- In Report Design Aid for the batch version.

  Use this option to ensure that the report is output to a CSV file every time *this* batch version is run.

  Select a batch version and in Report Design Aid, select the Export to CSV option in Print Setup. The batch version specifications will include information to export the output to a CSV file.

  The Export to CSV option is selected by the system at runtime. If a batch version is defined to export to CSV for every instance, you can clear the Export to CSV option at runtime when you do not want the batch version to export to a CSV file for a single submission.

- At runtime.

  Use this option to output batch versions to a CSV file *one time* only.

  When running the batch version locally, select the Export to CSV option when submitting the batch version.

  When running the batch version on the server, select Export to CSV (Comma Delimited) on the Document Setup tab of the Printer Selection form.

Before exporting report data to CSV, you should review the report to follow these recommendations:

- Set the horizontal grid alignment to 52 and select the snap to grid option.

  The default column width in spreadsheet programs is equivalent to 52 units in Report Design Aid. For best results, use these grid guideline so that each column included in the report template is equal to a column in the spreadsheet program.

- Ensure that no fields of the report overlap.

  If a data field overlaps into the next column, the data in the spreadsheet program displays in discrete columns. You can wrap the text in a cell once the data is exported to the spreadsheet program. Delete unused columns in the spreadsheet program and reformat information as needed.

- Align data fields vertically.

  If data fields are not aligned vertically, they display in separate rows in the spreadsheet program. If more than one data field with the same vertical and horizontal alignment displays in a column, only one of these fields displays in the CSV file. The first field output during the export occupies the cell in the spreadsheet program.

- Formats dates properly.

  Spreadsheet programs typically use the same date format used in the report.

- Use the Auto Format feature.

  After the report is exported cleanly, use the Auto Format feature in the spreadsheet program to further format the report.

- Countries that use a comma as a decimal marker.

  In these countries, the decimal separator is recognized as a comma when the report export. Tabs are stripped out instead of commas and a tab-separated file with a .txt extension is created.

  The information transfers as flat text, so totaling columns display only text. You must then set up totaling in the spreadsheet program.

When you export batch versions to CSV:

- A CSV file is created in the PrintQueue directory.
- A PDF file is created in the PrintQueue directory.
- The CSV file displays in a spreadsheet program such as Microsoft Excel or Lotus 123, which launches automatically when you run the batch version locally. When you run the bath version on a server, select View

CSV from Work With Servers (P986116) to launch the spreadsheet application and view the file. Only single spacing and portrait orientation is supported for CSV files. Drill-down links are ignored in CSV generation.

## Understanding OSA Interfaces

You can select to output to third-party software programs using OSA. OSA interfaces enable the third-party program to process and format the data concurrently. The OSA interface must be predefined; several interfaces may exist for one program, depending on the section types included in the report and the desired output.

OSA can use its own set of commands or an XML library. Because many software packages already use XML libraries for several functions, creating and using an XML library can simplify the interface.

Benefits of using OSA are:

- Eliminates the task of manually formatting output, as you must do with CSV output.
- Employs the processing power of the target software program.

## Understanding the Initialize Logical Printer Name System Function

You can use the Do Initialize Printer event to specify a printer for the system to use when the batch application processes. The Do Initialize Printer event is a report level event located on the File menu. Using this event, you can print the same report to different printers based on criteria that you define. The event rules located on this event are the first event rules processed at runtime. The event rules are also processed each time a subsystem trigger record is processed. The Initialize Logical Printer Name system function resolves and validates the printer name that you pass to it. The batch engine uses the printer name, if valid, to obtain a printer device context. Portions of this device context can be overridden when the appropriate flags in the report specifications are set.

The Initialize Logical Printer Name system function is ignored if placed on any event other than the Do Initialize Printer event. If you place this system function on a different event, the system generates a message in the jdedebug log.

## Prerequisite

Before defining printers, check out an existing report template.

## Defining Printers in Report Design Aid

From EnterpriseOne Life Cycle Tools, select Report Management (GH9111), Report Design Aid.

1. From the File menu, select and open a report template that is checked out.
2. From the File menu, select Print Setup.
3. On the Print Setup form, click the browse button immediately following the Printer Name field.
4. On the Printer Search & Select form, select the printer to use for the report, and click Select.

## Selecting Paper Types in Report Design Aid

Access the Print Setup form.

1. Select a predefined paper type from the drop-down list in the Size field.
2. If an appropriate paper type is not available, select Custom and indicate the paper width and height.

# Exporting to CSV in Report Design Aid

Access the Print Setup form.

1. Under Orientation, select Portrait.
2. Under OneWorld Printer, select Export to CSV and click OK.
3. Verify that no columns or fields in the report overlap.
4. From the Layout menu, select Grid Alignment.
5. On the Alignment Grid form, set the horizontal spacing to *52*.
6. Select the Snap to Grid option, and click OK.

   The system applies these settings to the entire report.

## See Also

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Design Aid*, "Viewing Properties for Report Sections, Fields, Columns, and Rows"

# CHAPTER 4

# Understanding Print Properties at Runtime

This chapter discusses:

- Batch version submission.
- Jobs submitted from the Microsoft Windows client.
- Printer information.
- The IFS on the iSeries.
- Printer selections at runtime.
- Paper type selections at runtime.
- Print orientation selections at runtime.
- Export to CSV option at runtime.
- Print settings in the jde.ini.

## Batch Version Submission

You must use a batch version to process a report. You can submit batch versions many ways:

- Locally from the Microsoft Windows client.

  The client immediately launches the batch engine process. You have the option to view the output on screen or send it to a printer.

- On the server from the Microsoft Windows client.

  The server can handle the batch processing more efficiently than the workstation. At the time the batch version is submitted to the server for processing, a message is sent to the server with the defined data selection, data sequencing and other information necessary to run the report, (such as report interconnect values and printer information).

  You can submit a batch version to the server using:

- The Batch Versions application on the Microsoft Windows client.
- RUNUBE from the command line on the Microsoft Windows client.
- Another report or interactive application.

  You must define the report interconnect in Report Design Aid. The system launches the child batch version from a batch application or interactive application submitted from either the Microsoft Windows client or the web client.

- The Submit Job application on the web client.

Batch versions can *only* be submitted to the server from the web client.

- A report task on the web client PeopleSoft EnterpriseOne Menu.

  You can add a report as a task on the PeopleSoft EnterpriseOne Menu for easy access.

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Batch Versions*, "Submitting Batch Versions".

# Jobs Submitted From the Microsoft Windows Client

You can submit batch jobs from the Microsoft Windows client in these ways:

- On Screen.

  This option generates a PDF file that is displayed through the Adobe Acrobat Reader. Adobe Acrobat Reader is launched by the system when the processing of the batch version is complete.

- To Printer.

  This option enables you to modify output options from the Printer Selection form.

- Export to CSV.

  This option generates both a CSV and a PDF file. The report is displayed through a CSV viewer, such as Excel. The CSV viewer is launched automatically by the system when processing of the batch version is complete.

- Export using OSA.

  This option exports the report to a third-party software application. The location of the output is determined by the OSA interface. For example, the PeopleSoft EnterpriseOne Extended Markup Language (XML) interface creates an XML file in the same location where the PDF and CSV outputs are stored. An OSA library created by the vendor of the third-party software to which you are exporting the report, might store the OSA output in a different location.

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Batch Versions*, "Submitting Batch Versions," Submitting Batch Versions From the Microsoft Windows Client.

# Printer Information

When you launch batch versions from the Microsoft Windows client, the printer information is stored in the pUBEDs structure and passed to the batch engine.

When you submit the batch version to the server, the printer information that is obtained from the Printer Selection form is stored in the F986110 as a BLOB (Binary Large Object).

When a batch job is submitted using report interconnects, the printer definition for the child report is inherited from the parent report. The inherited values are Printer name, Print Immediate, SavePDL, Paper Type and Printer flags, Number of Copies, and Paper Source. Export to CSV is not inherited. If the child report has a printer name defined in its specifications or a custom paper type defined, then these properties override the inherited values.

# The IFS on the iSeries

When you print a batch version on the iSeries server, the resulting report is sent to the integrated file system (IFS) in PDF. You can map to the reports using Microsoft Windows Explorer.

The default PrintQueue directory is located under the install directory on the server. If you want to create a custom PrintQueue directory, you need to define the PrintQueue using a valid IFS file name.

Under [NETWORK QUEUE], change *OutputDirectory =system*, where *system* is the location of the PrintQueue directory.

# Printer Selections at Runtime

When you submit a batch version to a printer, the system looks first for the printer defined in Report Design Aid, if no printer is defined, the system uses the default printer defined in the Printers application. The system determines a printer based on a hierarchical structure:

1. Printer defined at runtime.
2. Printer defined in Report Design Aid.

   The printer definition becomes part of the report specifications.
3. Printer defined in the Printer application using this hierarchy:
   a. Specific report defined for the user or role.
   b. Specific report defined for *PUBLIC.
   c. All reports defined for the user or role.
   d. All reports defined for *PUBLIC.

Depending on the printer that is selected, the system selects the corresponding Printer Definition Language on the Advance tab of the Printer Selection form at runtime.

When batch jobs are submitted using report interconnects, the child report inherits the printer definitions from the parent report. At print time, you can override the printer for the child report from the Work With Servers program (P986116) once the job has completed processing.

When you submit a job using RUNUBE, if you do not indicate a printer name through the command line parameters, the printer name stored in the specifications is used at print time; otherwise, the default printer is used. At print time, you can override the printer from the Work With Servers program when the job has completed processing.

# Paper Type Selections at Runtime

The paper type that you defined for the selected printer is used during job submission. From the Print Property tab on the Printer Selection form, you can change the paper type. Depending on the printer that you select, the Paper Type field is automatically populated by the system with the paper type that is defined for that printer.

When you define a custom paper size in Report Design Aid, you cannot change the paper type at runtime.

### See Also

Chapter 5, “Working with Report Printing Administration,” page 17

# Print Orientation Selections at Runtime

The paper orientation that you select in Report Design Aid is stored in the report specifications. Orientation from the specifications is displayed on the Print Property tab of the Printer Selection form at runtime. You can, however, change the orientation at runtime.

When you define a custom paper size in Report Design Aid, the orientation control is unavailable for selection on the Printer Selection form. In the case of line printers, the Characters per Inch (CPI), Characters per Page (CPP), Lines per Inch (LPI), and Lines per Page (LPP) options that you defined for the line printer determine the orientation.

# Export to CSV Option at Runtime

When you select the Export to CSV option in Report Design Aid, the definition is stored in the report specifications. This option is displayed in the Report Output Destination form at runtime. You can override the CSV option at runtime.

When you submit the batch version to the printer, you have the option of overriding the Export to CSV option on the Document Setup tab of the Printer Selection form. A CSV file and a PDF file are created in the PrintQueue directory when the CSV option is selected.

When you submit batch jobs to the server, and the Export to CSV option is defined in Report Design Aid, you have the opportunity to modify the option from the Work With Servers program (P986116).

When batch jobs are submitted using report interconnects, the child report *does not* inherit the export to CSV option from the parent. Therefore, you must select the CSV option in Report Design Aid to enable this option for both the child and parent reports.

# Print Settings in the jde.ini

The Print Immediate option and the Save PDL setting are defined in the initialization files.

### The Print Immediate Setting

You can define batch jobs to print immediately by modifying the Print Immediate flag in the initialization file. For the Microsoft Windows client, the Print Immediate flag is located in the jde.ini. For the web client, the Print Immediate flag is located in the jas.ini.

In the jde.ini, modify:

```
[NETWORK QUEUE SETTINGS]
PrintImmediate=TRUE/FALSE
```

In the jas.ini, modify:

```
[OWWEB]
PrintImmediate=TRUE/FALSE
```

When you define batch jobs to print immediately:

- The batch job automatically prints.

  You do not have to select the Print option on the Work With Servers program (P986116).

  At runtime, the Print Immediate option is selected by the system on the Document Setup tab on the Printer Selection form. You can override this setting at runtime.

- The child report inherits the definition from the parent report when batch jobs are submitted using report interconnects.

  In this case, you cannot override the print immediate option at runtime.

You can pass the Print Immediate option as an argument when using RUNUBE from the command line. In this case, the job automatically prints when the PrintImmediate option in the jde.ini is set to TRUE.

## The SavePDL Setting

Define the SavePDL option in the jde.ini on the enterprise server. The Printer Definition Language File setting is read by both the Microsoft Windows client and the web client.

When you set the SavePDL option to TRUE, the Printer Definition Language File option on the Document Setup tab of the Printer Selection form is selected by the system at runtime. You can, however, override this option at runtime.

In the jde.ini on the enterprise server, modify:

```
[NETWORK QUEUE SETTINGS]
SavePDL=TRUE/FALSE
```

When you set the SavePDL option to FALSE, you can select the option at runtime to save the intermediate temporary file created. The PDL file, when created, resides in the PrintQueue directory. When batch jobs are submitted using report interconnects, the child report inherits this definition from the parent report.

You have the option of modifying the Printer Definition Language File option at runtime and, depending on whether the option is selected or cleared, a PDF and a PDL file are created for that batch job only.

# CHAPTER 5

# Working with Report Printing Administration

This chapter provides an overview of report printing administration, and discusses how to:

- Work with the Printer application.
- Set up barcode fonts.
- Work with multiple code sets for PCL.
- Design reports to print on line printers.
- Print reports.

## Understanding Report Printing Administration

The Printer Definition application (P98616) provides a single point of entry for configuring printers. The application enables you to define printers for workstations and enterprise servers. These definitions reside in PeopleSoft EnterpriseOne tables that are maintained by the Printer Definition program.

PeopleSoft EnterpriseOne includes a number of predefined reports and batch versions, which you can use to meet your business requirements. If you want to make modifications to one of these predefined reports or batch versions, it is recommended that you copy the report or batch version, and modify the copy. In addition, you can create custom reports using Report Design Aid. The PeopleSoft EnterpriseOne batch engine generates these reports in PDF. You can view the PDF files using Adobe Acrobat Reader.

Reports must have a batch version before you can process the report. Batch versions do not require user interaction. You submit batch versions for processing and make selections at runtime. These selections include: data selection, data sequencing, location where the report processes, logging capabilities to monitor processing, and the printer on which the report prints.

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Design Aid*

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Batch Versions*

## Working With the Printer Application

This section provides overviews of the printer application and null pass-through print filters and discusses how to:

- Add printers.
- Define default printers.
- Modify printers.

- Copy printers.
- Delete printers.
- Delete paper types.
- Add null pass-through print filters.
- Search for incorrect printer records.

# Understanding the Printers Application

PeopleSoft EnterpriseOne provides a single application for defining system printers. This Printers application uses a director interface to guide you through the set up process. Instructions display on each form of the director to guide you through the printer addition process using the Printers application (P98616). From this director, you can add new printers, modify existing printers, and define default printers for a combination of users, roles, hosts, environments, and reports. You can also add and modify the paper types and custom conversion programs that the printers use.

To define a printer for the PeopleSoft EnterpriseOne system you must first add a printer. You must complete all of the fields that display on the director forms. Set up printers for each server platform that is used for processing reports in your enterprise. After printers are added, you must define a default printer.

## Platform Information

When defining server name and shared names for printers, consider these platform specific guidelines:

- iSeries:

  *library name/outqueue name*

  For the iSeries, the physical printer name must be the same as the outqueue name. If you use the default QGPL library to store the outqueues, enter only the outqueue name. This information must be entered in upper case.

  Example: `QutputQueue Name:DEVPRN1`

  If the outqueues reside in a library other than the default QGPL library, enter the library name and the outqueue name.

  Example:
  ```
  Library Name:QLIBRARY
  OutputQueue Name:DEVPRN1
  ```

  **Note.** When you qualify the outqueue name with the library name, you avoid possible name conflicts that might result in the submission of the report to an unexpected outqueue.

- Windows NT:

  *print server name**print shared name*

  Example:
  ```
  Print Server Name:corprts1
  Print Shared Name:devprn1
  ```

  For Windows NT, enter the name of the print server and the name of the printer. You cannot use spaces or special characters. This information must be entered in lower case. The system uses the print server name along with the print shared name to create the printer name.

- UNIX:

  *printer name* (no slashes)

  Example: `devprn16`

This information must be entered in lower case.

For printing reports to a non-network printer, leave the printer name field blank.

### Printer Definition Language

When defining the Printer Definition Language for printers, you might be presented with additional options based on the Printer Definition Language and platform type you select:

- When defining line printers, you must also define:
  - Characters per inch
  - Columns per page
  - Lines per inch
  - Lines per page

  **Note.** Use this formula to calculate the paper dimensions:

  Columns per page/Characters per inch = width in inches (85/10 = 8.5)

  Lines per page/Lines per inch = height in inches (66/6 = 11)

- When you define a line printer, the system disables the PostScript and PCL options. The system also disables the detail area at the bottom of the form. Any paper types selected are cleared.
- When you define a line printer that uses the iSeries platform, options appear within a box labeled iSeries Only. Use these fields to define the iSeries encoding that the printer supports:
  - ASCII Encoding
  - EBCDIC Encoding
- When you define a PostScript or PCL printer in combination with the iSeries platform, the ASCII Encoding option is automatically selected and the iSeries Only box is disabled.
- Only users with knowledge of building parameter strings for printers should use the custom option. The custom option uses an advanced feature of the Printers application. When you define a custom printer, a field appears beneath the Custom option. Enter the name of the conversion filter that you want to use in this field. You can add or modify conversion filters by selecting Advanced from the Form menu. The Advanced option is available only when the Custom option is selected. You can select an existing filter or add new filters on the Work With Conversion Programs form.

  See Chapter 5, "Working with Report Printing Administration," Understanding Null Pass-Through Print Filters, page 20.

### Paper Types

When defining printers, you can select from predefined paper types or add new paper types from the Form menu. You need to enter these definitions for new paper types:

- Paper Type

  The type of paper that the defined printer supports, such as legal, letter, and A4.

- Paper Height

  A value that indicates the height of the paper for the selected paper type.

- Paper Width

A value that indicates the width of the paper for the selected paper type.

- Unit of Measure

  The unit of measure used to indicate the paper height and width.

The system saves the new paper type and displays it on the Work With Paper Types form. When your definition is complete, the new paper type is available in the grid area of the Printer Setup form. All previous paper type selections are cleared and must be redefined.

### Default Printers

After adding a printer to the system, you must define the printer as the default printer. The printer can be defined as the default printer for a specific version of a report, for all versions of a report, or for all reports. You can also indicate that the printer is the default printer for a specific user or role, and a specific environment and host.

You must define the default printer as active for the system to recognize it as the default printer. Where multiple printers are configured using the same information, only one printer can be defined as active. If another printer is already defined as the active default, you must change the original default printer to inactive before making the new printer active. You can perform multiple status changes from the Work With Default Printers form.

### See Also

*PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: System Administration*, "Setting Up EnterpriseOne Printing"

## Understanding Null Pass-Through Print Filters

The null pass-through print filter enables you to send PDF documents directly to a print queue without converting it to a printer language format. Use null pass-through print filters when you define a custom Printer Definition Language for a printer.

You can select from available null pass-through print filters or create new filters. You must first select the Custom Printer Definition Language on the Printer Setup form and then select Advanced from the Form menu.

If you are making a copy of a conversion program, the Parameter String field is populated based on the filter that you selected on the Work With Conversion Programs form. Otherwise, the parameter string is entered automatically based on the host from which you are printing (for example ISERIES or HP9000) and the type of printer (postscript, PCL, or line). For example:

```
-s string_name
```

`-l` *library_name* `-f convertPDFToPS`

Where -s defines the string name, -l defines the library name (this value is the letter l, not the number 1), and -f defines the function name

# Forms Used to Add Printers

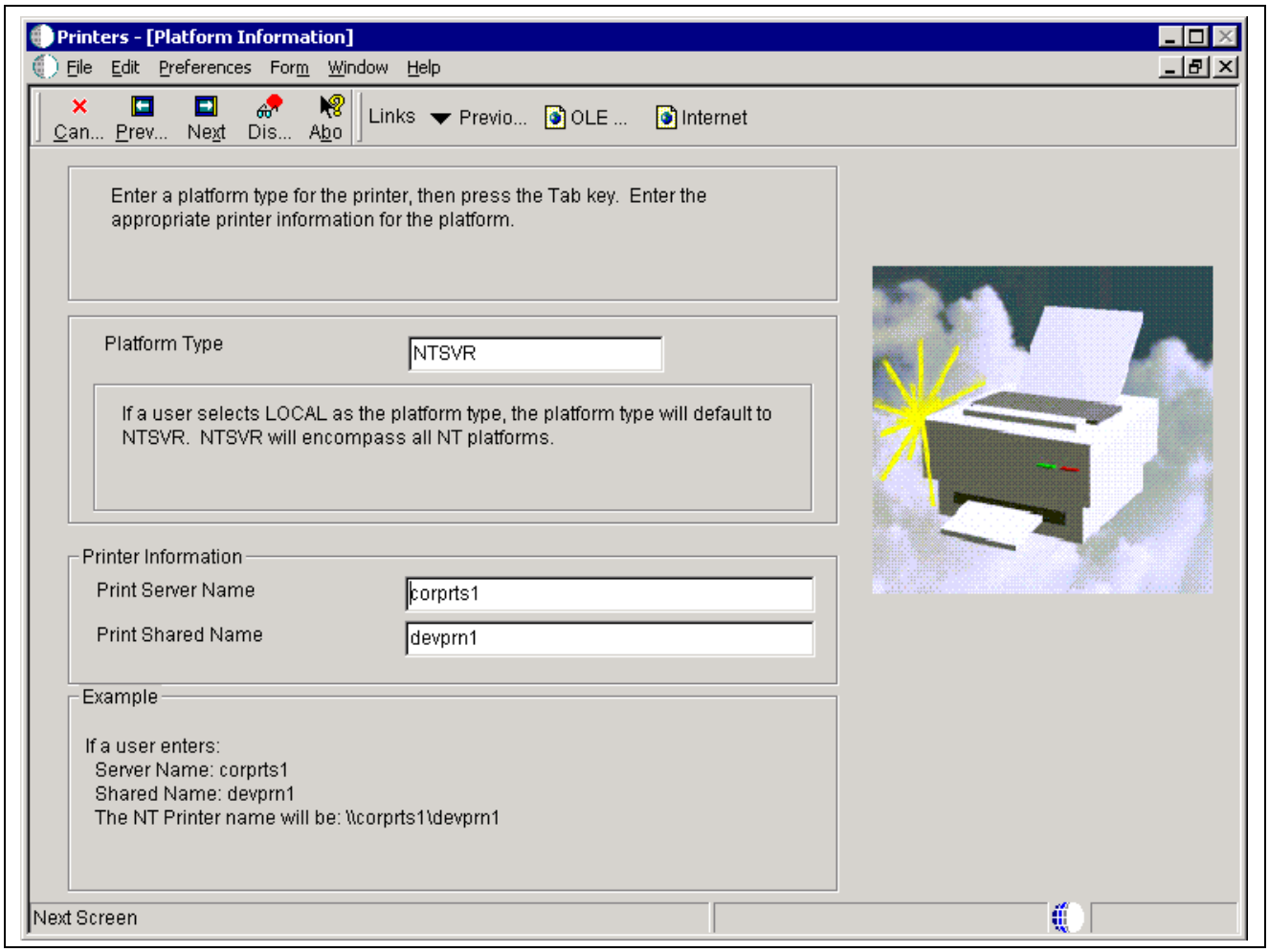
| Form Name | FormID | Navigation | Usage |
|---|---|---|---|
| Printers | W98616S | EnterpriseOne Life Cycle Tools, Report Management, Batch Processing Setup (GH9013), Printers | Add printers, modify printers, and define default printers |
| Printer Setup Director | W98616AC | Click Add Printer on the Printers form. | Review the printer tasks and begin the Printer Setup Director. |
| Platform Information | W98616V | Click Next on the Printer Setup Director form. | Enter platform type, printer server name, and print shared name. |
| Work With Printers | W98616Y | Click Modify Printer on the Printers form. | Select a printer to modify or copy. |
| Printer Setup | W98616AE | • Click Next on the Platform Information form when adding printers.<br>• Select a printer and click Select on the Work With Printers form when modifying printers. | Enter or modify the location and model of the printer, paper types, default paper type, and Printer Definition Language. |
| Work With Default Printers | W98616O | Click Define Default Printer on the Printers form. | Select a printer record. |
| Default Printer Revisions | W98616M | Click Select on the Work With Default Printers form. | Change the status of a printer. |
| Work With Conversion Programs | W98616I | Select Custom on the Details tab of the Printer Setup form and then select Advanced from the Form menu. | Select or add a conversion program. |
| Advanced Conversion Program | W98616J | Click Add on the Work With Conversion Programs form. | Enter a new conversion program name and parameter string. |

# Adding Printers

Access the Printers form.

### Platform Information

Access the Platform Information form.



Platform Information form

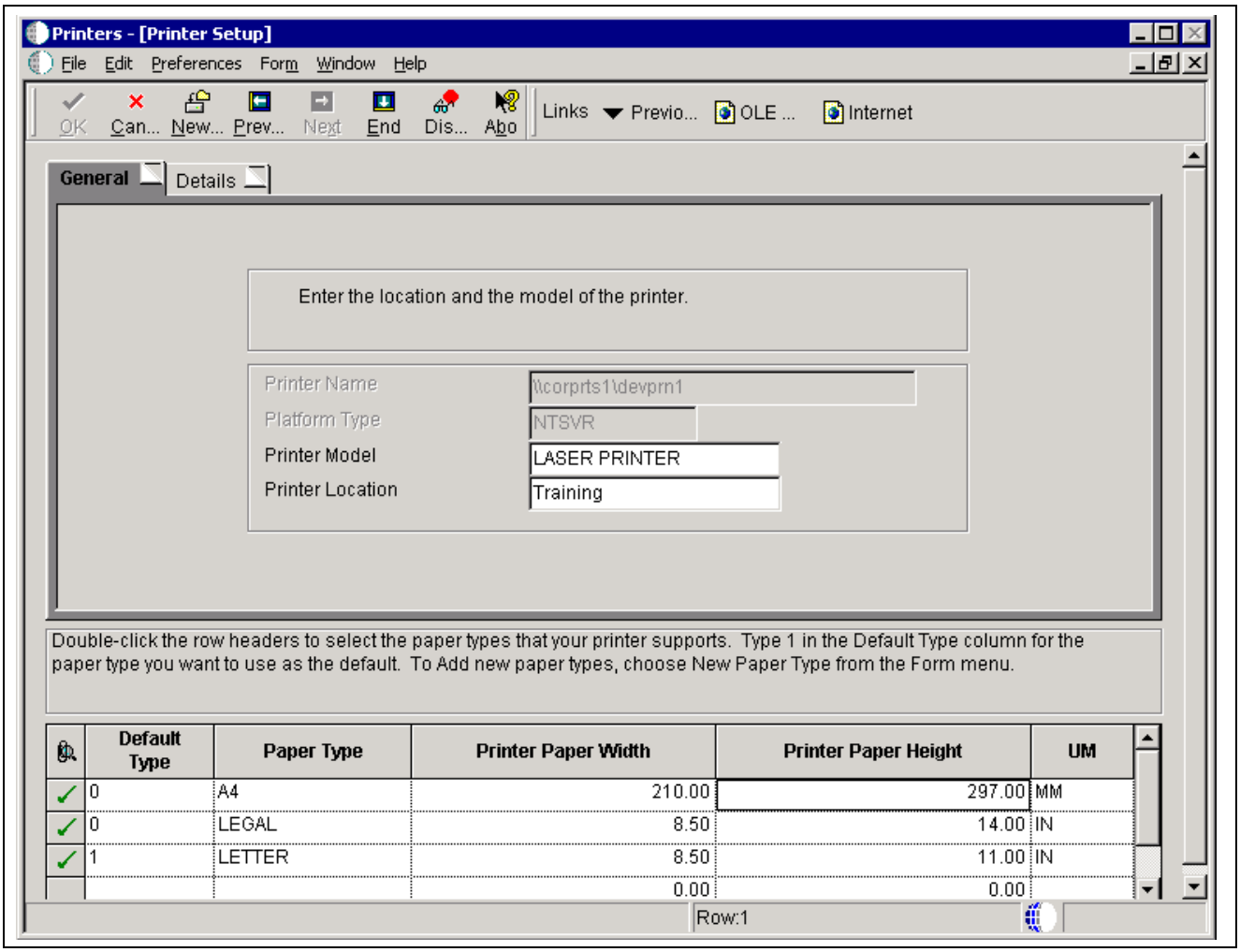
**Platform Type** The type of hardware on which the database resides.

**Print Server Name** The name of the machine that receives documents from clients

**Print Shared Name** The name of the printer to which the report will be sent.

## Printer Setup

Access the General tab on the Printer Setup form.




Printer Setup form

**Printer Name** The system populates this field based on the Print Server Name and the Print Shared Name that was entered on the Platform Information form. This information cannot be modified from this form.

**Platform Type** The system populates this field based on the Platform Type that was entered on the Platform Information form. This information cannot be modified from this form.

**Printer Model** The model of the printer.

**Printer Location** The physical location where the printer resides.

**Attachments** Check marks indicate that the paper type is selected for use with the defined printer.

**Default Type** A user defined code that indicates that the paper type is selected for the defined printer and whether the paper type is the default.

**Paper Type** The general size of paper that the defined printer supports, such as A4, legal, and letter size.

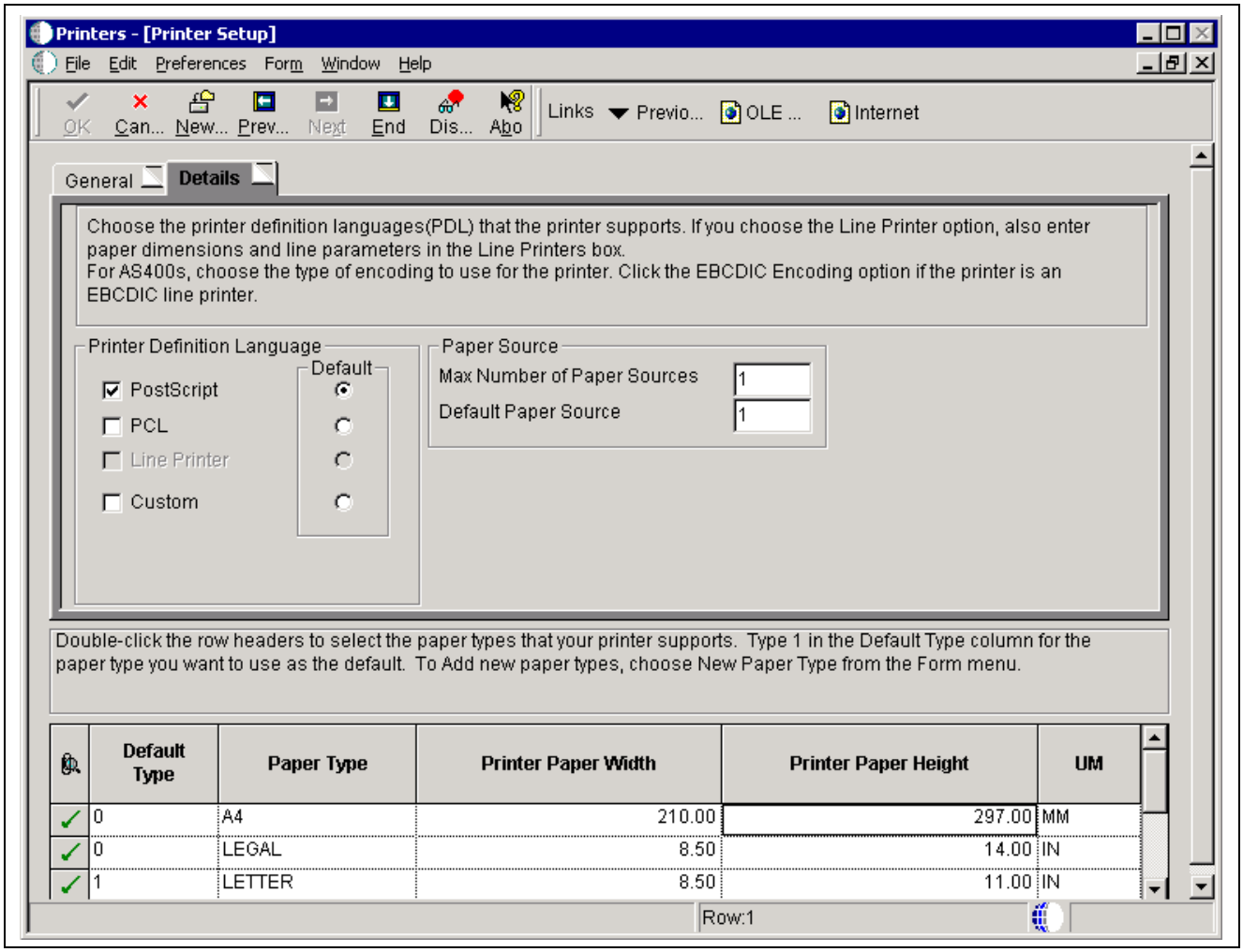
**Printer Paper Width** The value that indicates the width of the paper for the defined paper type. The value is displayed in the unit of measure defined in the UM field.

**Printer Paper Height** The value that indicates the height of the paper for the defined paper type. The value is displayed in the unit of measure defined in the UM field.

**UM** A user defined code that indicates the unit of measure that is used to define the width and height of the defined paper type.

## Printer Setup

Access the Details tab on the Printer Setup form.




Printer Setup form

**Printer Definition Language** The name of the Printer Definition Language (PDL) used by the defined printer. When the PostScript or PCL options are selected, the system disables the Line Printer option.

Multiple PDLs can be selected, but only one can be defined as the default. The PDL can be overridden when batch versions are submitted.

---

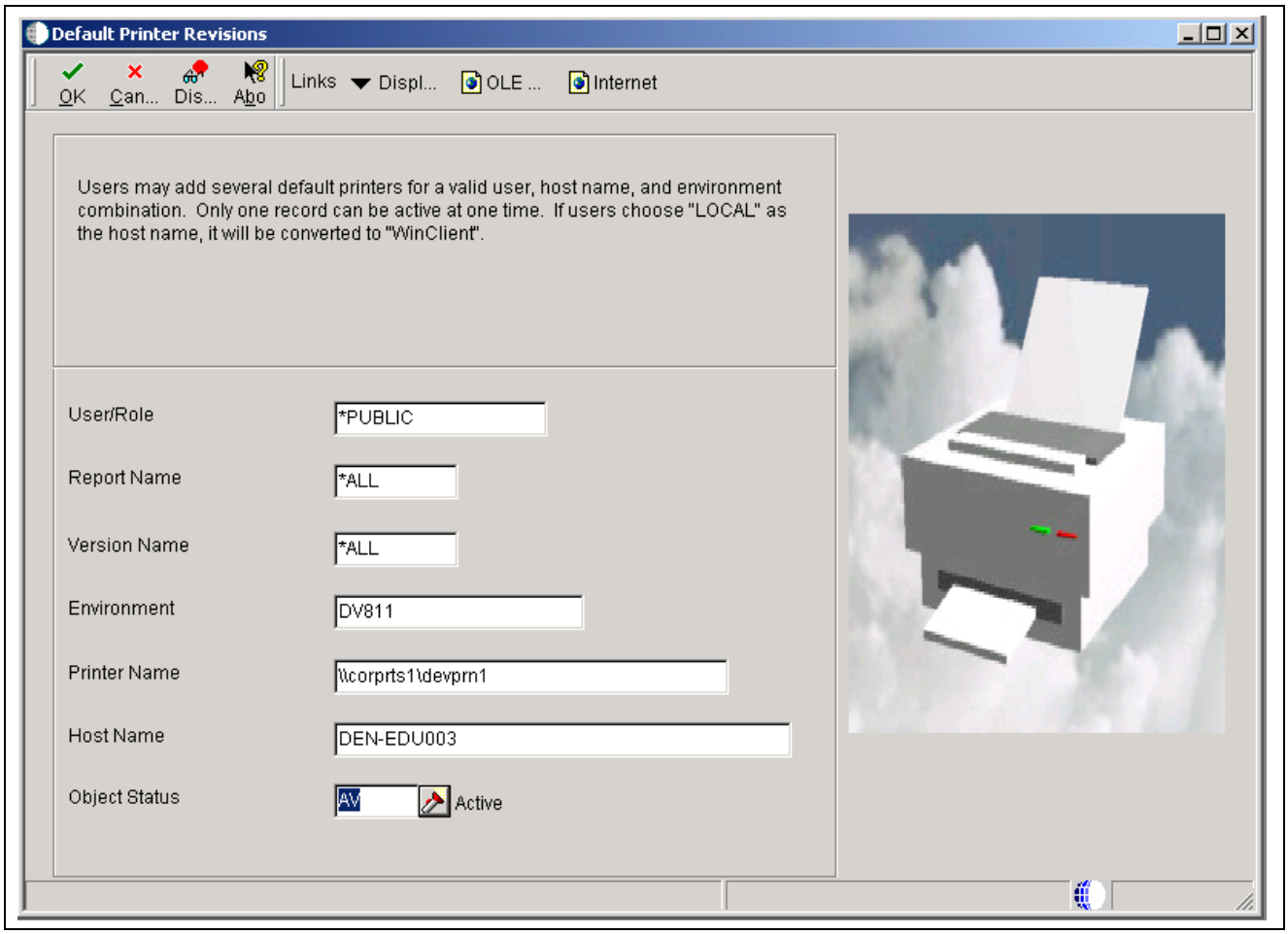
**Important!** The custom option uses an advanced feature of the Printers application. Only users with knowledge of building parameter strings for printers should use this option.

---

| | |
|---|---|
| **Maximum Number of Paper Sources** | The maximum number of paper trays available on the defined printer. Only available when the PostScript or Custom Printer Definition Language is selected. |
| **Default Paper Source** | The output tray to be used for a specific batch job. Only available when the PostScript or Custom Printer Definition Language is selected. |
| **Characters per Inch** | The number of characters per horizontal inch supported by the defined printer. Only available when the Line Printer PDL is selected. For an 8 1/2 x 11 inch page the characters per inch value is 10. |
| **Columns per Page** | The number of columns per page supported by the defined printer. Only available when the Line Printer PDL is selected. For an 8 1/2 x 11 inch page the columns per page value is 85. |
| **Line per Inch** | The number of lines per inch supported by the defined printer. Only available when the Line Printer PDL is selected. For an 8 1/2 x 11 inch page the lines per inch value is 6. |
| **Line per Page** | The number of lines per page supported by the defined printer. Only available when the Line Printer PDLe is selected. For an 8 1/2 x 11 inch page the lines per page value is 66. |
| **Printer Paper Width** | Populated by the system based on the columns per page value. Only appears when the Line Printer PDL is selected. |
| **Printer Paper Height** | Populated by the system based on the lines per inch value. Only appears when the Line Printer PDL is selected. |

## Defining Default Printers

Access the Default Printer Revisions form.




Default Printer Revisions form

**User/Role** The user ID or role with permissions to use the printer. *PUBLIC gives permissions to all users.

**Report Name** The name of the report that will use this printer. If the field is left blank, the default value is *ALL. *ALL allows all reports to use the printer.

**Version Name** The name of the batch version that will use this printer. If the field is left blank, the default value is *ALL. *ALL allows all batch versions to use the printer. If the Report Name is *ALL, the version name defaults to *ALL and is unavailable for input.

**Environment** Identifies the location of the report and batch version specifications. The system automatically enters the name of the environment that you are currently signed into. Change this information, if necessary.

**Printer Name** The name of the printer defined in the Printers application to be used as the default.

**Host Name** The name of the server that processes the defined batch versions. The visual assist displays the appropriate host names based on the printer name selected.

**Object Status** Indicates whether the default printer is active or inactive.

## Modifying Printers

Access the Work With Printers form.

1. Select the printer that you want to modify, and click Select.
2. On the Printer Setup form, modify the information for the printer as necessary.

   You cannot modify the printer name and platform type. If you select a line printer, the paper-type grid at the bottom of the form is disabled.

## Copying Printers

Access the Work With Printers form.

1. Select the printer that you want to copy, and click Copy.
2. On the Printer Setup form, complete fields as appropriate.
3. Select the Details tab and complete information as appropriate.

## Deleting Printers

Access the Work With Printers form.

1. Select a printer, or select multiple printers by holding down the CTRL key.
2. Click Delete.

   This task removes the printer definition.

## Deleting Paper Types

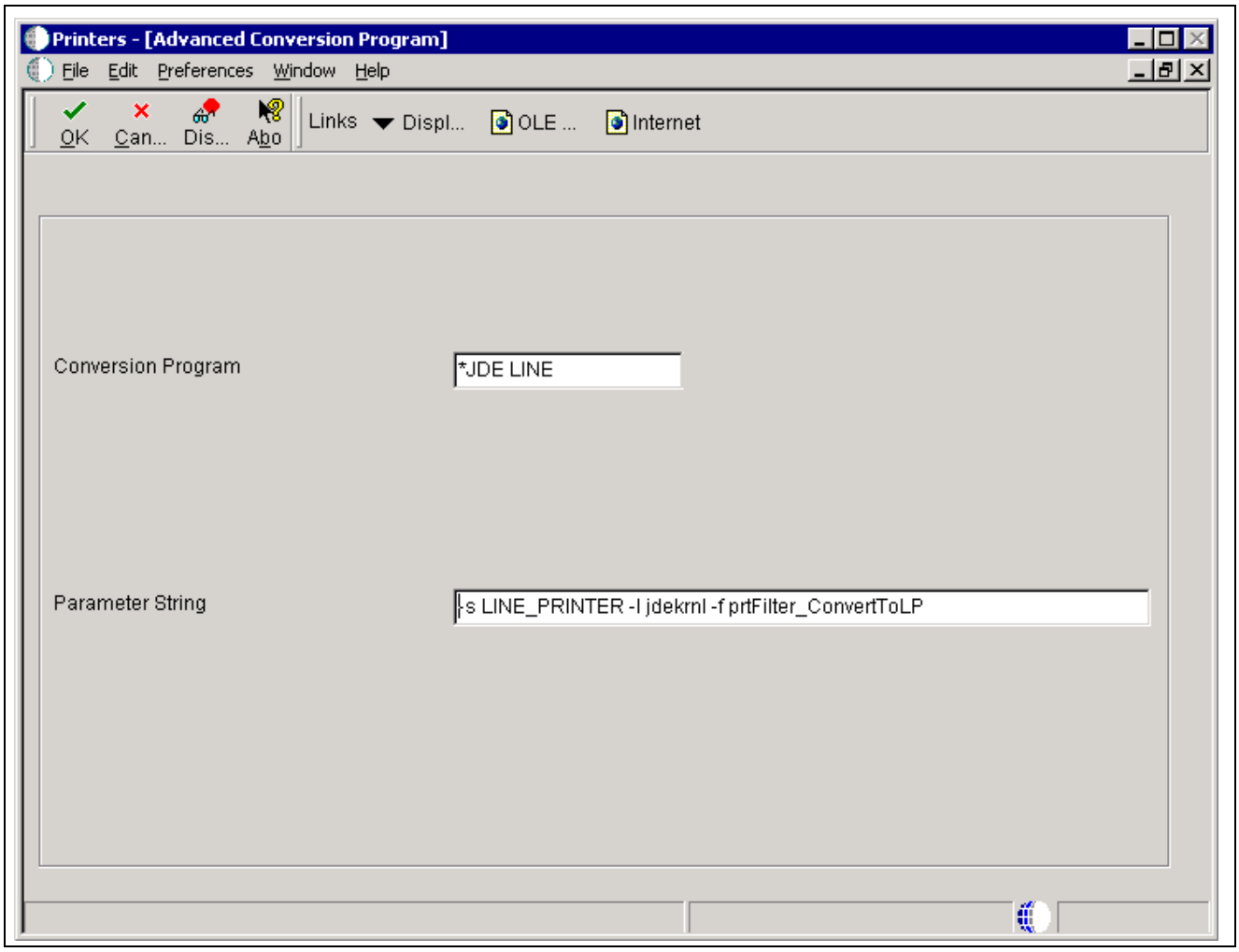
Access the Work With Printers form.

1. Select a printer and click Select.
2. On the Printer Setup form, select New Paper Type from the Form menu.
3. On the Work With Paper Types form, select a paper type and click Delete.
4. On Confirm Delete, click OK.

   The paper type that you deleted no longer displays in the detail area.

## Adding Null Pass-through Print Filters

Access the Advanced Conversion Program form.



Advanced Conversion Program form

**Conversion Program** The name of the conversion program (null pass-through print filter). Use the visual assist to select a valid program.

**Parameter String** The parameter value passed to the conversion program. This value is populated by the system based on the conversion program selected. The parameter string is dependent on the type of printer, and the hardware and software platform being used.

## Searching for Incorrect Printer Records

Use this batch process to search the Printer Capability table (F986163) and list printer records that are incomplete, or that contain incorrect printer information. This report can help you identify incomplete printing records.

From EnterpriseOne Life Cycle Tools, select Report Management (GH9111), Batch Versions.

1. On the Work With Batch Versions - Available Versions form, enter *R9861602* in the Batch Application field and click Find.
2. Run the XJDE0001 version.

The report lists any printer definition records that might not be complete or that are erroneous and need correction.

3. Using the report, locate the incomplete printer record and correct it.

   See Chapter 5, "Working with Report Printing Administration," Modifying Printers, page 27.

# Setting Up Barcode Fonts

This section provides an overview of barcode fonts and discusses how to:

- Set up printers to use barcode fonts.
- Modify barcode printer information.
- Copy barcode printer information for new printers.
- Delete barcode support information from printers.

## Understanding Barcode Fonts

PeopleSoft EnterpriseOne supports the use of the BC C39 3 to 1 Medium barcode font, and includes this font with the software. After you set up the printers, you can assign a printer to use a barcode font for use with reports. This section describes how to define a printer to support the barcode font BC C39.

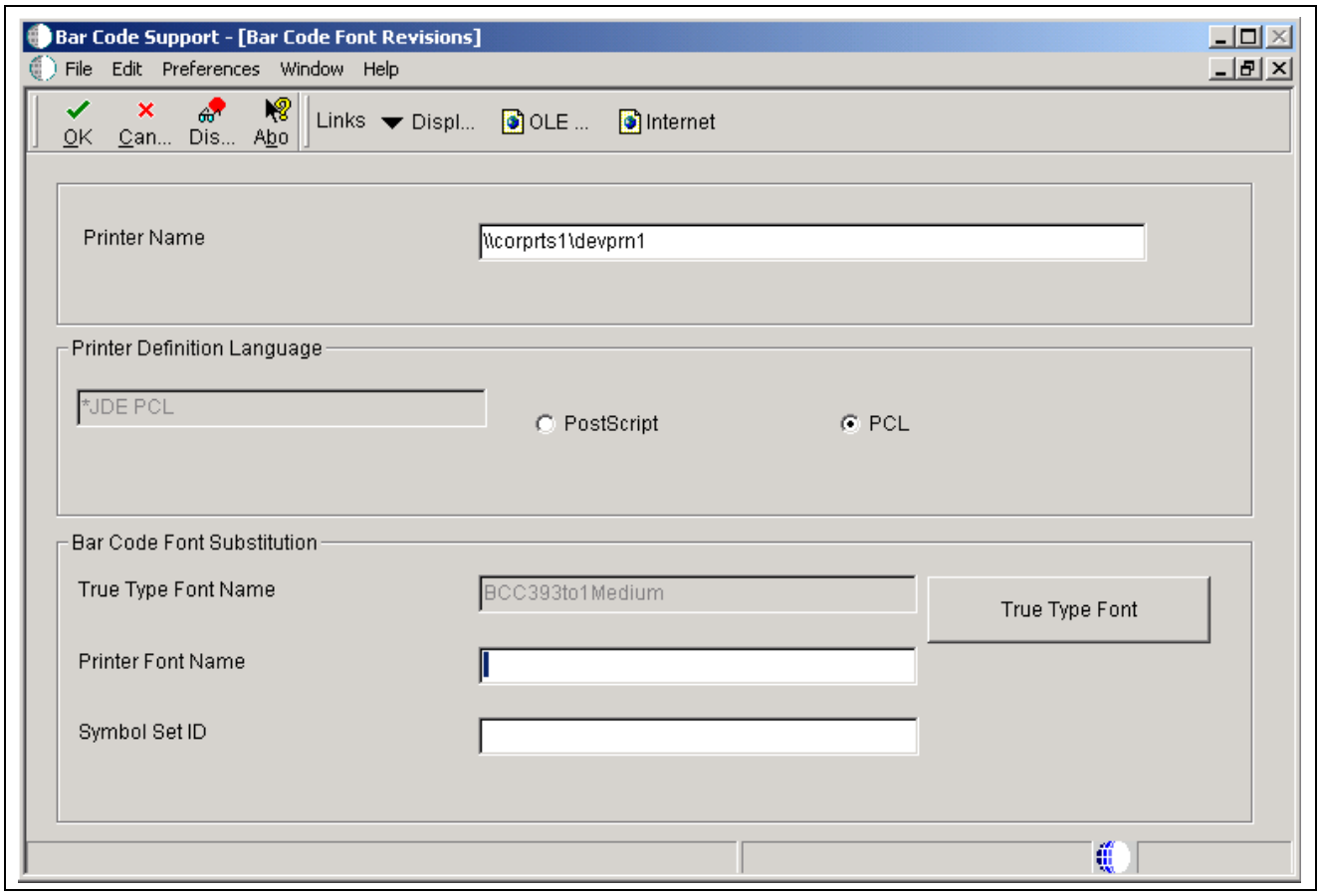
**Note.** Printers that support barcode fonts must use either the PostScript or PCL printer definition languages.

## Forms Used to Set Up Printers to Use Barcode Fonts

| Form Name | FormID | Navigation | Usage |
|---|---|---|---|
| Work With Bar Code Font | W986166A | EnterpriseOne Life Cycle Tools, Report Management, Batch Processing Setup (GH9013), Bar Code Support | Modify or add printers defined to support barcodes. |
| Bar Code Support Revisions | W986166B | Click Add on the Work With Bar Code Font form. | Enter the printer name, printer definition language, True Type font, font name, and symbol set ID to be used for barcodes. |

## Setting Up Printers to Use Barcode Fonts

Access the Bar Code Support Revisions form.




Bar Code Font Revisions form

**Printer Name** The name of the printer to be used for printing barcode fonts.

**Printer Definition Language** The printer definition language used by the printer selected. Options include PostScript and PCL

**True Type Font Name** The name of the barcode font to be used. Click the True Type Font button to select the *BC C39 3 to 1 Medium* barcode font. The bar code true type font must reside in the Microsoft Windows font directory.

**Printer Font Name** The name of the printer font to be used.

**Symbol Set ID** A value that defines the character and character mapping for a specific symbol set. Contact the PCL printer font vendor to obtain this information. This option is available for the PCL printer definition language only.

## Modifying Barcode Printer Information

Access the Work With Bar Code Font form.

1. Select the printer for which you want to modify information, and click Select.
2. On the Bar Code Font Revisions form, modify the information as appropriate.

## Copying Barcode Printer Information for New Printers

Access the Work With Bar Code Font form.

1. Select the printer that you want to copy and click Copy.
2. On the Bar Code Font Revisions form, enter the name of the new printer.
3. Modify information as appropriate.

### Deleting Barcode Support Information From Printers

Access the Work With Bar Code Font form.

1. Select the printer that you want to delete, and click Delete.
2. On the Confirm Delete form, click OK.

## Understanding Multiple Code Sets for PCL

This section discusses:

- Multiple code sets for PCL printing.
- The order of precedence for PCL printing.

### Multiple Code Sets for PCL Printing

The PeopleSoft EnterpriseOne batch engine generates reports using Unicode data. This means that a single report can contain many different kinds of characters representing multiple languages. However, since PCL printers do not directly support Unicode data, the PCL print filter must translate the report data and tell the printer how to display it.

There are two values that control the translation and display of individual characters in PCL:

- Code page
- Symbol set

You can configure your system to use either a single combination of code page and symbol set, or multiple combinations per report to support an international environment. One of the settings that controls this functionality is the `PRTPCLSymbolSet` setting under the [UBE] heading of the jde.ini and the jas.ini. Consider these possibilities:

- The PRTPCLSymbolSet definition is missing or is set to `PRTPCLSymbolSet = *AUTO`.

  The PCL print filter automatically analyzes the report data to determine which code page and symbol set values to use to correctly display the characters in the report. The print filter then sends the appropriate values to the printer as it is printing.

- The PRTPCLSymbolSet definition is set to `PRTPCLSymbolSet = *NONE`.

  A single symbol set is derived based on the code page used to generate the report. This combination of symbol set and code page is used for the entire document.

- The PRTPCLSymbolSet definition is set to `PRTPCLSymbolSet =`*XXX*, where *XXX* represents the desired symbol set.

  The defined symbol set is used for all PCL printing.

## The Order of Precedence for PCL Printing

The order of precedence determines which code sets or fonts are used when there are multiple print settings defined. PCL printing has two orders of precedence; the first determines code page and the second determines symbol set. The print filter proceeds down the list until a valid value is derived, and then it uses that value.

### Code Page Order of Precedence

When a print job is sent to a PCL printer, the PCL print filter:

- Checks the F986166 table to determine if the report contains barcode information.

  If the report includes barcode information, it prints the report using a bar code character set.
- Checks to determine if multiple code pages are enabled (PRTPCLSymbolSet=*AUTO).

  If multiple code pages are enabled, it analyzes text strings from the report to determine the code page.
- Checks the PRTLocalCodeSet setting and uses the code page indicated by the setting.
- Uses the code page associated with the user's language preference.

### Symbol Set Order of Precedence

After the code set is determined, the PCL print filter uses these rules to determine the symbol set:

- Checks to determine if a symbol set is specified in the F986166 table.

  If a symbol set is specified, it uses that symbol set.
- Checks the PRTPCLSymbolSet setting.

  If a specific symbol set is indicated, it uses that set.
- Checks the symbol set code associated with the code page.

  The print filter selects a symbol set that matches the current code page.

# Designing Reports to Print on Line Printers

This section provides overviews of reports designed to print on line printers and defining remote iSeries printers to print multiple copes of reports, lists the prerequisite, and discusses how to:

- Modify reports to print on line printers.
- Print multiple copies of reports to remote iSeries line printers.

## Understanding Reports Designed to Print on Line Printers

When you design reports to print on line printers, you must follow certain guidelines to ensure that the information contained in the reports print successfully. These guidelines include font family, font size, grid spacing, the width of the fields on the report, paper dimensions, and line parameters:

- Modify the vertical grid alignment.
- Select a fixed-pitch font.

The Courier New font is a fixed-pitch font and provides the best results; however, you can use other fixed-pitch fonts. For example, for reports that contain text in Japanese, you should use the fixed-pitch version of the MS-Gothic font.

- Modify the font size.
- Modify the field width.

  Since font properties affect the size of the report fields, you might need to adjust their width. You need to override version specifications for a section, specifically the section layout, before you can modify field widths.
- Apply the font selections to the entire report.
- Align fields.

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Report Design Aid*, "Modifying the Appearance of Report Objects".

## Understanding Defining Remote iSeries Line Printers to Print Multiple Copies of Reports

If the output queue for an iSeries line printer does not support printing multiple copies, the Display Options parameter can be modified to send multiple copies of documents to the remote printer. A system administrator must perform this task, and only for remote output queues.

## Prerequisite

Before modifying reports to print on line printers, ensure that you perform one of these steps:

- Create a batch version of the report that will be sent only to line printers. Do not make the modifications in the report template because then the report data might not display properly on other printer platforms.
- Check out an existing batch version for modifying to print to line printers.

## Modifying Reports to Print on Line Printers

Select a batch version that is checked out and launch Report Design Aid.

1. From the Layout menu, select Grid Alignment.
2. On the Alignment Grid form, modify the value in the Vertical field to *16*and click OK.
3. From the File menu, select Report Properties.
4. On the Properties form, select the Font/Color tab, and change the font to *Courier New.*
5. Change the font size to *10.*
6. Select the Apply settings to all objects option and click OK.
7. Widen fields as necessary to provide enough room for the data to display on the report.
8. If some data fields do not properly align, align the fields using the Align option on the Layout menu
9. Save the version.

## Printing Multiple Copies of Reports to Remote iSeries Line Printers

To print multiple copies to remote iSeries line printers:

1. End the remote writer to which the output queue is connected.
2. Use the Change Output Queue (CHGOUTQ) command to modify the Display Options (DSPOPT) parameter to contain the value *XAIX*.
3. Restart the remote writer.

# Printing Reports

This section discusses:

- Batch versions at submission.
- Batch versions processed on the server.
- Batch versions processed locally on the Microsoft Windows client.
- Print-time characteristics.
- Print settings for batch versions.

## Batch Versions at Submission

When you submit batch versions, the batch engine uses a device context to generate a PDF file. This device context includes information such as page size and the printable area of a page. The system generates this information from the printer tables for all platforms.

In PeopleSoft EnterpriseOne, you have the option of viewing the PDF output using Adobe Acrobat Reader, or sending the report directly to a printer. You can also print the report from Adobe Acrobat Reader. When you send the report to a printer, the system uses a conversion filter to transform the PDF file into one of three Page Description Language (PDL) formats: PCL, PostScript, or line-printer text. These language formats depend on the type of printer printing the report.

The batch engine uses a logical path to determine to which printer to send reports. If the first method does not return a valid printer name, the batch engine uses the subsequent method.

When you submit batch versions:

1. The batch process triggers the Do Initialize Printer event defined in Report Design Aid.

   If this process retrieves a valid printer name, other processes are ignored.
2. You override the default printer name at the time that the report is submitted.

   If you override the default printer with a valid printer name, further processes are ignored.
3. The report specifications pass a printer name to the batch process.

   If this process retrieves a valid printer name, the next process is ignored.
4. The system uses the F98616 table to determine a valid default printer based on the current user, the environment that the user is signed into, and the host that processes the report.

## Batch Versions Processed on the Server

When you submit batch versions to the server, the engine prompts you for a printer name. Valid printer names must have been previously defined by the system administrator. The server automatically creates a PDF file using the settings associated with the selected printer, unless event rules override those printer settings. You can affect how the report prints by modifying settings on the Printer Selection form, such as the printer name, page orientation, PDL, and paper type.

When you view the report from the Microsoft Windows client, the system copies the PDF file from the server to the local directory, E811\PrintQueue on the workstation.

When you view the report from the web client, the system copies the PDF file from the server to a temp directory on the workstation. The temp directory is defined in the jas.ini. PDF files are deleted when you sign off of the web client.

In the jas.ini, modify:

```
[JDENET]
tempFileDir=<temp directory location>
```

---

**Note.** When you are using an iSeries platform, the PDF file is stored in the integrated file system (IFS).

---

See Chapter 4, "Understanding Print Properties at Runtime," The IFS on the iSeries, page 13.

When you run batch versions, you have the option of activating logging capabilities from the Advanced form. If you process batch versions locally from the Microsoft Windows client, the workstation stores the log file in the E811\PrintQueue directory.

If you process batch versions on the server, either from the Microsoft Windows client or the web client, the enterprise server stores the log file in the server PrintQueue directory.

## Batch Versions Processed Locally From the Microsoft Windows Client

When you run batch versions locally from a Microsoft Windows client and view the output on the screen, the engine tries to connect to the printer defined in Report Design Aid. If the engine cannot connect, or if there is no printer defined, the engine uses the default printer from the printer tables. Using the settings that it retrieves, the engine creates a PDF file and displays the report through Adobe Acrobat Reader. The PDF file is stored in the E811\PrintQueue directory on the workstation.

When you run batch versions locally on the Microsoft Windows client and send the output to a printer, the engine displays the Printer Selection form. This form presents you with options to change the printer, page orientation, PDL, paper type, and so on. The initial printer displayed on the Printer Selection form is the one defined in Report Design Aid, or the default printer if one was not defined. The engine connects to the printer displayed on the printer form and retrieves the associated settings. Using these settings, the engine creates a PDF file, converts the PDF into a PDL file using the conversion filter, and sends the PDL file to the defined printer.

See *PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Development Tools: Batch Versions*, "Submitting Batch Versions," Submitting Batch Versions From the Microsoft Windows Client.

# Print-Time Characteristics

At print time, you can override the printer defined for a report. This is different than overriding the printer when you submit the batch version. At runtime, you can select any valid enterprise printer. At print time, however, you can *only* override the printer with another printer that supports the same platform, PDL, and paper type as the original printer. This is because the batch engine has already created the PDF version of the report and has imbedded the platform, PDL, and paper type information into the PDF file.

# Print Settings for Batch Versions

On the Microsoft Windows client, the workstation jde.ini settings control whether reports print immediately and whether the system saves the output after processing the report:

```
[NETWORK QUEUE SETTINGS]
PrintImmediate=TRUE/FALSE
SaveOutput=TRUE/FALSE
```

This table describes the jde.ini settings:

| Setting | Description |
|---|---|
| PrintImmediate | Specifies whether the system automatically prints the report after processing is complete. Values are:<br>TRUE.<br>The system processes the report on the server, generates a PDF file, converts the PDF to the appropriate PDL for the defined printer, and then prints the report.<br>FALSE.<br>The system processes the report on the server, but does not automatically print the report. Users must access the Work with Servers application to manually print the report. |
| SaveOutput | Specifies whether the system saves or deletes the output after the user views or prints the job. Values are:<br>TRUE.<br>The system saves the output after it has been viewed or printed.<br>FALSE.<br>The system deletes the output after it has been viewed or printed. |

The jas.ini settings control whether reports print immediately and whether the system saves the output after processing the report on the web client.

```
[OWWEB]
PrintImmediate=TRUE/FALSE
KeepUBE=TRUE/FALSE
```

This table describes the jas.ini settings:

| Setting | Description |
| --- | --- |
| PrintImmediate | Specifies whether the system automatically prints the report after processing is complete. Values are:<br><br>TRUE.<br><br>The system processes the report on the server, generates a PDF file, converts the PDF to the appropriate PDL for the defined printer, and then prints the report.<br><br>FALSE.<br><br>The system processes the report on the server, but does not automatically print the report. Users must access View Job Status to manually print the report. |
| KeepUBE | Specifies whether the system saves or deletes the output after the user views or prints the job. Values are:<br><br>TRUE.<br><br>The system saves the output after it has been viewed or printed.<br><br>FALSE.<br><br>The system deletes the output after it has been viewed or printed. |

# CHAPTER 6

# Working with Output Stream Access

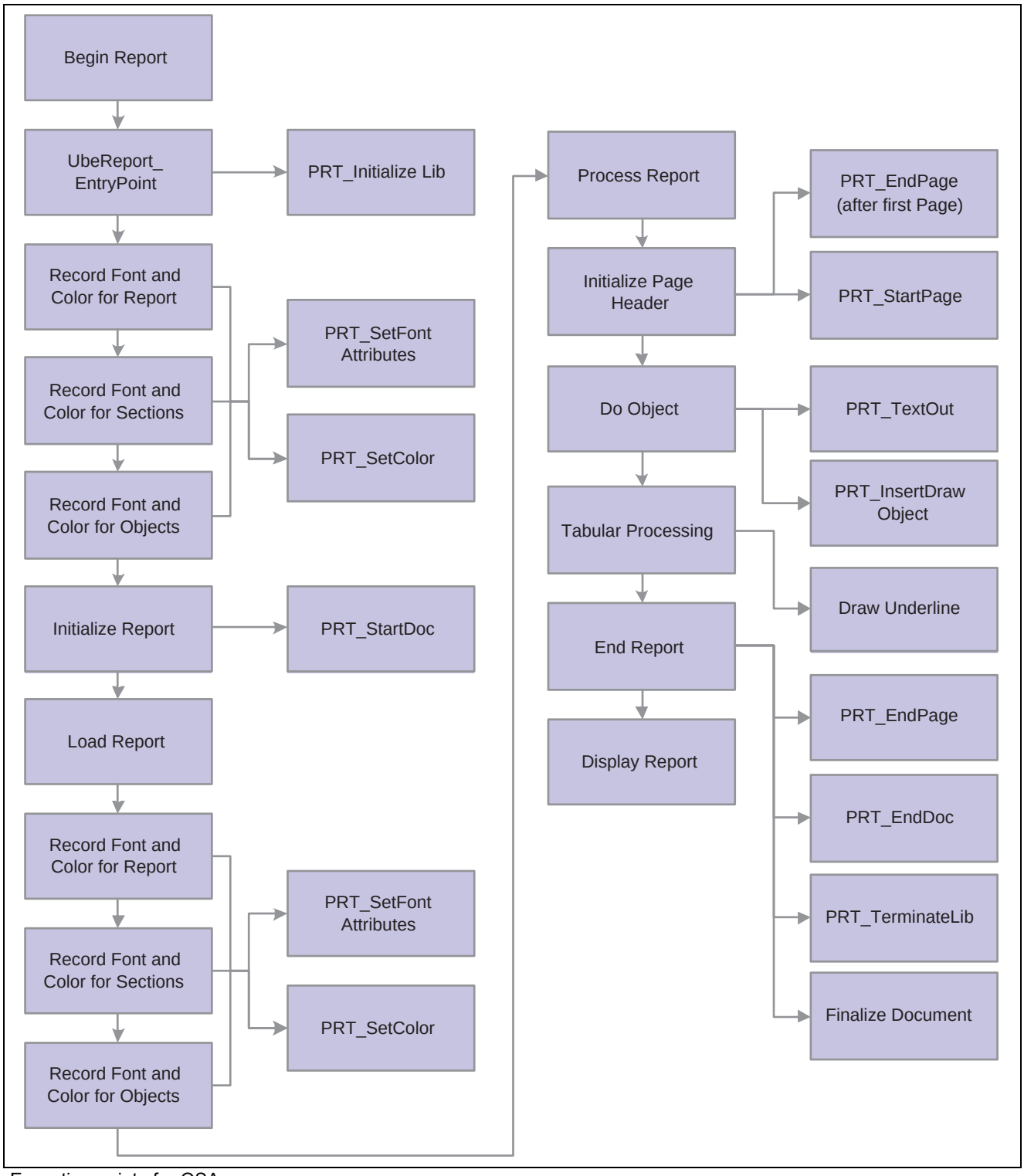
This chapter provides an overview of Output Stream Access (OSA) and discusses how to:

- Create OSA libraries.
- Create and associate OSA interfaces.

## Understanding Output Stream Access

The system processes the components of batch applications in a specific order. At different points during this processing, you can trigger an event through the OSA interface.

This diagram illustrates the OSA execution points:




Execution points for OSA

At each of these execution points, with the exception of PRT_InitializeLib and PRT_TerminateLib, you can call an OSA function. You can create functions or you can use existing XML libraries and their functions.

# Creating OSA Libraries

This section discusses:

- OSA libraries.
- Function parameters.
- OSA documents.
- Include files.
- File locations and names.
- OSASample source code.

## OSA Libraries

An OSA library is a collection of OSA functions; OSA functions must be included in a library before you can use them. You can create new functions and libraries to satisfy your business requirements.

PeopleSoft EnterpriseOne includes an OSA library named OSASample. This OSASample library, along with its source code, serves as an example of how to create an OSA library. The following reference information is provided for developers who need to create their own libraries.

### Function Signatures

OSA functions are called using the function pointers defined in the JDEOSA file. Therefore, OSA functions should be defined using the same parameters and return values, as in this example set of function prototypes:

```
void MyStartDoc (POSA_REPORT_INFO);
void MySetFont (POSA_REPORT_INFO, POSA_FONT_INFO);
void MySetColor (POSA_REPORT_INFO, unsigned long int);
void MyStartPage (POSA_REPORT_INFO);
void MyTextOut (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyDrawObject (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyUnderline (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyEndPage (POSA_REPORT_INFO, POSA_LINK_INFO, unsigned long);
void MyEndDoc (POSA_REPORT_INFO, POSA_PAGEOF_INFO, unsigned long);
void MyFinalize (POSA_REPORT_INFO);
void MyStartDoc (POSA_REPORT_INFO);
void MySetFont (POSA_REPORT_INFO, POSA_FONT_INFO);
void MySetColor (POSA_REPORT_INFO, unsigned long int);
void MyStartPage (POSA_REPORT_INFO);
void MyTextOut (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyDrawObject (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyUnderline (POSA_REPORT_INFO, POSA_OBJECT_INFO);
void MyEndPage (POSA_REPORT_INFO, POSA_LINK_INFO, unsigned long);
void MyEndDoc (POSA_REPORT_INFO, POSA_PAGEOF_INFO, unsigned long);
void MyFinalize (POSA_REPORT_INFO);
```

## Function Parameters

Function parameters define:

- Start document parameters
- Set font parameters
- Set color parameters
- Start page parameters
- Text out parameters
- Draw object parameters
- Draw underline parameters
- End page parameters
- End document parameters
- Finalize Document Parameters

### Defining Start Document Parameters

The OSA function that is associated with the Start Document execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo
OSA_REPORT_INFO *pOSAReportInfo
```

### Defining Set Font Parameters

The OSA function that is associated with the Set Font execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
OSA_FONT_INFO *pOSAFontInfo
```

### Defining Set Color Parameters

The OSA function that is associated with the Set Color execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
unsigned long int zColorRef
```

### Defining Start Page Parameters

The OSA function that is associated with the Start Page execution point is called by defining this parameter:

```
OSA_REPORT_INFO *pOSAReportInfo
```

### Defining Text Out Parameters

The OSA function that is associated with the Text Out execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
OSA_OBJECT_INFO *pOSAObjectInfo
```

### Defining Insert Draw Object Parameters

The OSA function that is associated with the Insert Draw Object execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
OSA_OBJECT_INFO *pOSAObjectInfo
```

### Defining Draw Underline Parameters

The OSA function that is associated with the Draw Underline execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
OSA_OBJECT_INFO *pOSAObjectInfo
```

### Defining End Page Parameters

The OSA function that is associated with the End Page execution point is called by defining these parameters:

```
OSA_REPORT_INFO pOSAReportInfo,
OSA_LINK_INFO *pOSALinkInfo,
unsigned long ulNumberOfLinks
```

### Defining End Document Parameters

The OSA function that is associated with the End Document execution point is called by defining these parameters:

```
OSA_REPORT_INFO *pOSAReportInfo,
OSA_PAGEOF_INFO *pOSAPageOfInfo,
unsigned long ulNumberOfPageOf
```

### Defining Finalize Document Parameters

The OSA function that is associated with the Finalize Document execution point is called by defining this parameter:

```
OSA_REPORT_INFO *pOSAReportInfo
```

## OSA Documents

Within the OSAReportInfo structure, the szOSAFileName member enables external applications to specify the name of a file created by OSA functions. A member of the same name is added to the UBEVar structure. After the End Document execution point has been processed, any value that exists in the OSAReportInfo member for szOSAFileName is copied to the corresponding UBEVar member. When a job has finished processing, the UBEVar structure is updated into the F986110 table for that job.

## Include Files

The structure and function definitions required for functions interfacing through OSA are contained in the JDEOSA.H file, which is located in the system\include directory. The current contents of this file are:

```
/******************************************************
Header File Description
* JDEOSA.H Header file to support Output Stream Access functions
******************************************************
Copyright (c) 1999 - 2004
PeopleSoft, Inc
*
This material is proprietary to PeopleSoft, Inc.
All rights reserved. The methods and techniques described herein are considered
trade secrets and confidential. Reproduction or distribution, in whole or in
```

```
part, is forbidden except by express written permission of PeopleSoft, Inc.
*******************************************************/
#ifndef JDEOSA_H
#define JDEOSA_H

/** Link Information Structure **/
struct tagOSA_LINK_INFO
{
float      fLowerLeftHorizontal;
float      fLowerLeftVertical;
float      fUpperRightHorizontal;
float      fUpperRightVertical;
JCHAR      szApplication[11];
JCHAR      szForm[11];
JCHAR      *szParms;
};
typedef struct tagOSA_LINK_INFO  OSA_LINK_INFO, * POSA_LINK_INFO;
/** Font Information **/
struct tagOSA_FONT_INFO
{
long int     lfHeight;
long int     lfWidth;
long int     lfEscapement;
long int     lfOrientation;
long int     lfWeight;
BYTE       lfItalic;
BYTE       lfUnderline;
BYTE       lfStrikeOut;
BYTE      lfCharSet;
BYTE      lfOutPrecision;
BYTE      lfClipPrecision;
BYTE      lfQuality;
BYTE      lfPitchAndFamily;
JCHAR      lfFaceName[32];
unsigned short   nPointSize;
JCHAR      szAdobeFontName[100];
};
typedef struct tagOSA_FONT_INFO  OSA_FONT_INFO, * POSA_FONT_INFO;
/** Item Information **/
struct tagOSA_ITEM_INFO
{
unsigned long   ulOccurenceCount;
unsigned long   ulRecordFetchCount;
unsigned long   ulNumPDFLines;
unsigned short  nReprinting;
unsigned short  nUnderlineThickness;
unsigned short  nUnderlineMargin;
unsigned long int ColorRef;
OSA_FONT_INFO   zFontInfo;
unsigned short  nDisplayStyle;
```

```
int       bPrintMetaData
float      fObjectHorizontalPosition;
float      fObjectVerticalPosition;
float      fObjectEndingHorizontalPosition;
float      fObjectEndingVerticalPosition;
float      fValueHorizontalPosition;
float      fValueVerticalPosition;
float      fValueEndingHorizontalPosition;
float      fValueEndingVerticalPosition;
JCHAR      *szValue;
JCHAR      *szFullText;
};
typedef struct tagOSA_ITEM_INFO  OSA_ITEM_INFO, * POSA_ITEM_INFO;
/** Object Information **/
struct tagOSA_OBJECT_INFO
{
JCHAR      szDataDictionaryAlias[41];
JCHAR      szObjectName[31];
unsigned long   idObject;
unsigned long   idSection;
unsigned long   idRow;
JCHAR      szObjectType[3];
unsigned long   idLength;
unsigned long   idEverestType;
JCHAR      cDataType;
OSA_ITEM_INFO   zOSAItemInfo;
void      *pOSASectionInfo;
void      *pExternalDataPointer;
JCHAR      szFutureUse[256];
};
typedef struct tagOSA_OBJECT_INFO  OSA_OBJECT_INFO, * POSA_OBJECT_INFO;
/** Section Information Structure **/
struct tagOSA_SECTION_INFO
{
JCHAR      *szSectionName;
JCHAR      szSectionType[50];
short      nSectionType;
JCHAR      szBusinessViewName[11];
unsigned long   idSection;
unsigned long   idParentSection;
unsigned long   ulNumberOfObjects;
unsigned long   ulRecordFetchCount;
OSA_OBJECT_INFO  *pOSAObjectInfo;
void      *pExternalDataPointer;
JCHAR      szFutureUse[256];
};
typedef struct tagOSA_SECTION_INFO  OSA_SECTION_INFO, * POSA_SECTION_INFO;
/*** Report Information Structure ***/
struct tagOSA_REPORT_INFO
{
```

```
JCHAR      szReport[11];
JCHAR      szVersion[11];
JCHAR      szMachineKey[16];
JCHAR      szEnhv[11];
JCHAR      szRole[11]
JCHAR      szUser[21];
JCHAR      szHostName[80]
JCHAR      szOneWorldRelease[11];
JCHAR      szReportTime[12];
JCHAR      szDateToday[11];
unsigned int    nLocalCodePage;
unsigned int    nRemoteCodePage;
int       nLocalOperatingSystem;
int       nRemoteOperatingSystem;
JCHAR      szPrinter[256];
unsigned long   ulPageSizeVertical;
unsigned long   ulPageSizeHorizontal;
unsigned long   ulNumberOfCopies;
unsigned long   ulPaperSource;
unsigned short   nPageOrientation;
unsigned short   nPrinterLinesPerInch;
unsigned short   nPrinterCharactersPerInch;
unsigned short   nPrinterDefaultFontSize;
JCHAR      szPDLProgram[11];
JCHAR      szDecimalString[2];
JCHAR      cThousandsSeparator;
JCHAR      szDateFormat[5];
JCHAR      cDateSeparator;
JCHAR      szLanguage[3]
JCHAR      *szReportTitle;
JCHAR      szCompanyName[31];
unsigned long   ulJobNum;
unsigned long   ulCurrentPageNumber;
unsigned long   ulActualCurrentPageNumber;
JCHAR      szUBEFileName[300];
JCHAR      szOSAFileName[256];
JCHAR      szOSAClientFileName[31]
unsigned long   ulNumberOfSections;
OSA_SECTION_INFO  *pOSASectionInfo;
void       *pExternalDataPointer;
unsigned short   *pnLogMessageSeverity;
JCHAR      szLogMessage[256];
JCHAR      szFutureUse[256];
};
typedef struct tagOSA_REPORT_INFO  OSA_REPORT_INFO, * POSA_REPORT_INFO;
/** Execution Point Identification Numbers **/
#define  OSA_EXPN_START_DOC  1
#define  OSA_EXPN_SET_FONT   2
#define  OSA_EXPN_SET_COLOR  3
#define  OSA_EXPN_START_PAGE  4
```

```
#define  OSA_EXPN_TEXT_OUT   5
#define OSA_EXPN_DRAW_OBJECT 6
#define  OSA_EXPN_UNDERLINE  7
#define  OSA_EXPN_END_PAGE   8
#define  OSA_EXPN_END_DOC   9
#define  OSA_EXPN_FINALIZE_DOC 10

/** OSA Function Prototypes **/
#if defined (_WIN32)
  #define  OSACDECL _cdecl
#else
  #define  OSACDECL
#endif

typedef void (OSACDECL *FP_OSA_START_DOC)    (POSA_REPORT_INFO);
typedef void (OSACDECL *FP_OSA_SET_FONT)    (POSA_REPORT_INFO,POSA_FONT_INFO);
typedef void (OSACDECL *FP_OSA_SET_COLOR)   (POSA_REPORT_INFO,unsigned long int);
typedef void (OSACDECL *FP_OSA_START_PAGE)   (POSA_REPORT_INFO);
typedef void (OSACDECL *FP_OSA_TEXT_OUT)    (POSA_REPORT_INFO,POSA_OBJECT_INFO);
typedef void (OSACDECL *FP_OSA_DRAW_OBJECT) (POSA_REPORT_INFO,POSA_OBJECT_INFO);
typedef void (OSACDECL *FP_OSA_UNDERLINE)    (POSA_REPORT_INFO,POSA_OBJECT_INFO);
typedef void (OSACDECL *FP_OSA_END_PAGE)    (POSA_REPORT_INFO,POSA_LINK_⇒
INFO,unsigned long);
typedef void (OSACDECL *FP_OSA_END_DOC)    (POSA_REPORT_INFO);
typedef void (OSACDECL *FP_OSA_FINALIZE_DOC)   (POSA_REPORT_INFO);
#endif
```

## File Locations and Names

The Universal Batch Engine (UBE) performs these steps to load an OSA Library:

1. If the library name, as defined in the F986169 table, contains a period (.), the UBE ignores the period and any subsequent characters.
2. The UBE adds prefixes, extensions, or both according to the platform on which the UBE is executing.

This table illustrates the prefixes and extensions that are added to the platform on which the UBE is executing:

| PLATFORM | EXTENDED LIBRARY NAME |
|---|---|
| PC | libname + .dll |
| HPUX | lib + libname + .sl |
| AIX, SUN | lib + libname + .so |
| iSeries | libname |

The UBE passes the resulting library name to the LoadLibrary function, which uses a standard search strategy to locate the requested library.

### OSA File Names

The OSA files are generated in the same location as the PDF files, which is the PrintQueue directory. They also follow a similar naming convention as the PDF files, except for the extension. For example, the OSASample output has the extension, .osa.

## OSASample Source Code

OSASample includes three components:

- OSAStruct.h
- OSASample.h
- OSASample.c

### OSAStruct.h

This example shows OSAStruct.h source code:

```
#ifndef   OSASAMPLE_DEF_HPP
#define   OSASAMPLE_DEF_HPP
#define   chAmpersand '&'
#define   chOpenAngle '<'
#define   chCloseAngle '>'
#define   chDoubleQuote ''
#define   PAGEOF_TYPE TP
typedef struct tagOSASAMPLE_STRUCT
{
unsigned short  nCount;
FILE  *fpOutput;
} OSASAMPLE_STRUCT, *POSASAMPLE_STRUCT;
#endif
```

### OSASample.h

This example shows OSASample.h source code:

```
#ifndef __OSASAMPLE_H__
#define __OSASAMPLE_H__
#include <string.h>
#include <assert.h>
#include <stdio.h>
#include <jdeosa.h>
#if defined (_WIN32)
#undef CDECL
#define CDECL _cdecl
#if defined(IAMOSASAMPLE)
#define APIEXPORT _declspec(dllexport)
#else
#define APIEXPORT _declspec(dllimport)
#endif
#else
#define CDECL
```

```
#define APIEXPORT
#endif
#define CLASSEXPORT APIEXPORT
#undef EXTERNC
#if defined(__cplusplus)
#define EXTERNC extern C
#else
#define EXTERNC
#endif
EXTERNC APIEXPORT void CDECL OSASample_StartDoc(POSA_REPORT_INFO
                        pOSAReportInfo);
EXTERNC APIEXPORT void CDECL OSASample_SetFont(POSA_REPORT_INFO
       pOSAReportInfo, POSA_FONT_INFO pOSAFontInfo);
EXTERNC APIEXPORT void CDECL OSASample_SetColor(POSA_REPORT_INFO
       pOSAReportInfo, unsigned long int zColorRef);
EXTERNC APIEXPORT void CDECL OSASample_EndDoc(POSA_REPORT_INFO
                        pOSAReportInfo,
POSA_PAGEOF_INFO pOSAPageofInfo,
unsigned long ulNumberOfStructs);
EXTERNC APIEXPORT void CDECL OSASample_StartPage(POSA_REPORT_INFO
pOSAReportInfo);
EXTERNC APIEXPORT void CDECL OSASample_EndPage(POSA_REPORT_INFO
pOSAReportInfo,
POSA_LINK_INFO pOsaLinkInfo,
unsigned long ulNumberOfLinks);
EXTERNC APIEXPORT void CDECL OSASample_TextOut(POSA_REPORT_INFO
 pOSAReportInfo, POSA_OBJECT_INFO pOSAObjectInfo);
EXTERNC APIEXPORT void CDECL OSASample_DrawObject(POSA_REPORT_INFO
 pOSAReportInfo, POSA_OBJECT_INFO pOSAObjectInfo);
EXTERNC APIEXPORT void CDECL OSASample_DrawUnderLine(POSA_REPORT_INFO
 pOSAReportInfo, POSA_OBJECT_INFO pOSAObjectInfo);
EXTERNC APIEXPORT void CDECL OSASample_FinalizeDoc(POSA_REPORT_INFO
 pOSAReportInfo);
#endif
```

## OSASAMPLE.c

This example shows OSASample.c source code:

```
#include OSASample.h
#include OSAStruct.h
/*-------------------------------------------------------------------------
* Function Name: OSA_ReportInfoOut
* Parameters: OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* OSA_REPORT_INFO * Pointer to Report Info structure
* Exceptions: None
* Return Value: None
* Description: Output data from the Report Info structure
*------------------------------------------------------------------------- */
void OSA_ReportInfoOut(POSASAMPLE_STRUCT pOSAStruct, POSA_REPORT_INFO pOSAReport⇒
Info)
```

```
{
/* Check for valid parameter values. If pointers are void, return. */
if (!pOSAStruct || !pOSAStruct->fpOutput || !pOSAReportInfo)
{
return;
}
/* Print values to the output file */
fprintf(pOSAStruct->fpOutput, ****** REPORT INFO ******\n\n);
fprintf(pOSAStruct->fpOutput, Report: %s\n, pOSAReportInfo->szReport);
fprintf(pOSAStruct->fpOutput, Version: %s\n, pOSAReportInfo->szVersion);
fprintf(pOSAStruct->fpOutput, MachineKey: %s\n, pOSAReportInfo->
                        szMachineKey);
fprintf(pOSAStruct->fpOutput, Environment: %s\n, pOSAReportInfo->szEnhv);
fprintf(pOSAStruct->fpOutput, User: %s\n, pOSAReportInfo->szUser);
fprintf(pOSAStruct->fpOutput, Release: %s\n, pOSAReportInfo->
                       szOneWorldRelease);
fprintf(pOSAStruct->fpOutput, Time: %s\n, pOSAReportInfo->szReportTime);
fprintf(pOSAStruct->fpOutput, Date: %s\n, pOSAReportInfo->szDateToday);
fprintf(pOSAStruct->fpOutput, Local Code Page: %d\n, pOSAReportInfo->
                         nLocalCodePage);
fprintf(pOSAStruct->fpOutput, Remote Code Page: %d\n, pOSAReportInfo->
                          nRemoteCodePage);
fprintf(pOSAStruct->fpOutput, Local Operating System: %d\n,
             pOSAReportInfo->nLocalOperatingSystem);
fprintf(pOSAStruct->fpOutput, Remote Operating System: %d\n,
pOSAReportInfo->nRemoteOperatingSystem);
fprintf(pOSAStruct->fpOutput, Printer: %s\n, pOSAReportInfo->szPrinter);
fprintf(pOSAStruct->fpOutput, Page Height: %d\n,
pOSAReportInfo->ulPageSizeVertical);
fprintf(pOSAStruct->fpOutput, Page Width: %d\n,
pOSAReportInfo->ulPageSizeHorizontal);
fprintf(pOSAStruct->fpOutput, Number Of Copies: %d\n,
pOSAReportInfo->ulNumberOfCopies);
fprintf(pOSAStruct->fpOutput, Paper Source: %d\n,
pOSAReportInfo->ulPaperSource);
fprintf(pOSAStruct->fpOutput, Orientation: %d\n,
pOSAReportInfo->nPageOrientation);
fprintf(pOSAStruct->fpOutput, Lines Per Inch: %d\n,
pOSAReportInfo->nPrinterLinesPerInch);
fprintf(pOSAStruct->fpOutput, Default Font Size: %d\n,
pOSAReportInfo->nPrinterDefaultFontSize);
fprintf(pOSAStruct->fpOutput, Printer Type: %s\n,
pOSAReportInfo->szPDLProgram);
fprintf(pOSAStruct->fpOutput, Decimal Separator: %s\n,
pOSAReportInfo->szDecimalString);
fprintf(pOSAStruct->fpOutput, Thousands Separator: %c\n,
pOSAReportInfo->cThousandsSeparator);
fprintf(pOSAStruct->fpOutput, Date Format: %s\n, pOSAReportInfo->
szDateFormat);
fprintf(pOSAStruct->fpOutput, Date Separator: %c\n, pOSAReportInfo->
```

```
cDateSeparator);
fprintf(pOSAStruct->fpOutput, Report Title: %s\n, pOSAReportInfo->
szReportTitle);
fprintf(pOSAStruct->fpOutput, Company Name: %s\n, pOSAReportInfo->
szCompanyName);
fprintf(pOSAStruct->fpOutput, Job Number: %d\n, pOSAReportInfo->ulJobNum);
fprintf(pOSAStruct->fpOutput, Current Page Number: %d\n,
pOSAReportInfo->ulCurrentPageNumber);
fprintf(pOSAStruct->fpOutput, Actual Page Number: %d\n,
pOSAReportInfo->ulActualCurrentPageNumber);
fprintf(pOSAStruct->fpOutput, UBE File Name: %s\n, pOSAReportInfo->
szUBEFileName);
fprintf(pOSAStruct->fpOutput, OSA File Name: %s\n, pOSAReportInfo->
szOSAFileName);
fprintf(pOSAStruct->fpOutput, Number of Sections: %d\n, pOSAReportInfo->
ulNumberOfSections);
fprintf(pOSAStruct->fpOutput, \n****** END REPORT INFO ******\n);
return;
}
/*-------------------------------------------------------------------------
* Function Name: OSA_SectionInfoOut
* Parameters: OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* OSA_SECTION_INFO * Pointer to Section Info structure
* Exceptions: None
* Return Value: None
* Description: Output data from the Section Info structure
*------------------------------------------------------------------------- */
void OSA_SectionInfoOut(POSASAMPLE_STRUCT pOSAStruct, POSA_SECTION_INFO
pOSASectionInfo)
{
/* Check for valid parameter values. If pointers are void, return. */
if (!pOSAStruct || !pOSAStruct->fpOutput || !pOSASectionInfo)
{
return;
}
fprintf(pOSAStruct->fpOutput, \n\t****** SECTION INFO ******\n\n);
fprintf(pOSAStruct->fpOutput, \tSection Name: %s\n, pOSASectionInfo->
szSectionName);
fprintf(pOSAStruct->fpOutput, \tSection Type: %s\n, pOSASectionInfo->
szSectionType);
fprintf(pOSAStruct->fpOutput, \tBusiness View Name: %s\n, pOSASectionInfo->
szBusinessViewName);
fprintf(pOSAStruct->fpOutput, \tSection ID: %d\n, pOSASectionInfo->
idSection);
fprintf(pOSAStruct->fpOutput, \tParent Section ID: %d\n, pOSASectionInfo->
idParentSection);
fprintf(pOSAStruct->fpOutput, \tNumber of Objects: %d\n, pOSASectionInfo->
ulNumberOfObjects);
fprintf(pOSAStruct->fpOutput, \tRecord Fetch Count: %d\n, pOSASectionInfo->
ulRecordFetchCount);
```

```
fprintf(pOSAStruct->fpOutput, \n\t****** END SECTION INFO ******\n);
return;
}
/*-------------------------------------------------------------------------
* Function Name: OSA_ObjectInfoOut
* Parameters: OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* OSA_OBJECT_INFO * Pointer to Object Info structure
* unsigned short int Flag to control output of Item Info
* Exceptions: None
* Return Value: None
* Description: Output data from the Object Info and Item Info structures
*------------------------------------------------------------------------- */
void OSA_ObjectInfoOut(POSASAMPLE_STRUCT pOSAStruct,
POSA_OBJECT_INFO pOSAObjectInfo,
unsigned short int nPrintItemInfo)
{
/* Check for valid parameter values. If pointers are void, return. */
if (!pOSAStruct || !pOSAStruct->fpOutput || !pOSAObjectInfo)
{
return;
}
fprintf(pOSAStruct->fpOutput, \n\t\t****** OBJECT INFO ******\n\n);
fprintf(pOSAStruct->fpOutput, \t\tData Dictionary Item: %s\n,
pOSAObjectInfo->szDataDictionaryAlias);
fprintf(pOSAStruct->fpOutput, \t\tObject Name: %s\n, pOSAObjectInfo->
szObjectName);
fprintf(pOSAStruct->fpOutput, \t\tObject ID: %d\n, pOSAObjectInfo->idObject);
fprintf(pOSAStruct->fpOutput, \t\tSection ID: %d\n, pOSAObjectInfo->
idSection);
fprintf(pOSAStruct->fpOutput, \t\tRow ID: %d\n, pOSAObjectInfo->idRow);
fprintf(pOSAStruct->fpOutput, \t\tObject Type: %s\n, pOSAObjectInfo->
szObjectType);
fprintf(pOSAStruct->fpOutput, \t\tObject Length: %d\n, pOSAObjectInfo->
nLength);
fprintf(pOSAStruct->fpOutput, \t\tOneWorld Data Type: %d\n, pOSAObjectInfo->
idEverestType);
fprintf(pOSAStruct->fpOutput, \t\tGeneral Data Type: %c\n, pOSAObjectInfo->
cDataType);
fprintf(pOSAStruct->fpOutput, \n\t\t****** END OBJECT INFO ******\n);
/* Only output Item Info if the parameter indicates to do so */
if (nPrintItemInfo)
{
POSA_ITEM_INFO pOSAItemInfo = &(pOSAObjectInfo->zOSAItemInfo);
fprintf(pOSAStruct->fpOutput, \n\t\t****** ITEM INFO ******\n\n);
fprintf(pOSAStruct->fpOutput, \t\tOccurence Count: %d\n, pOSAItemInfo->
ulOccurenceCount);
fprintf(pOSAStruct->fpOutput, \t\tRecord Fetch Count: %d\n, pOSAItemInfo->
ulRecordFetchCount);
fprintf(pOSAStruct->fpOutput, \t\tNumber Of Lines: %d\n, pOSAItemInfo->
ulNumPDFLines);
```

```
fprintf(pOSAStruct->fpOutput, \t\tReprinting: %d\n, pOSAItemInfo->
nReprinting);
fprintf(pOSAStruct->fpOutput, \t\tUnderline Thickness: %d\n, pOSAItemInfo->
nUnderlineThickness);
fprintf(pOSAStruct->fpOutput, \t\tUnderline Margin: %d\n, pOSAItemInfo->
nUnderlineMargin);
fprintf(pOSAStruct->fpOutput, \t\tColor Reference: %d\n, pOSAItemInfo->
ColorRef);
fprintf(pOSAStruct->fpOutput, \t\tFont Face Name: %s\n, pOSAItemInfo->
zFontInfo.lfFaceName);
fprintf(pOSAStruct->fpOutput, \t\tFont Point Size: %d\n, pOSAItemInfo->
zFontInfo.nPointSize);
fprintf(pOSAStruct->fpOutput, \t\tAdobe Font Name: %s\n, pOSAItemInfo->
zFontInfo.szAdobeFontName);
fprintf(pOSAStruct->fpOutput, \t\tDisplay Style: %d\n, pOSAItemInfo->
nDisplayStyle);
fprintf(pOSAStruct->fpOutput, \t\tObject Start X: %f\n, pOSAItemInfo->
fObjectHorizontalPosition);
fprintf(pOSAStruct->fpOutput, \t\tObject Start Y: %f\n, pOSAItemInfo->
fObjectVerticalPosition);
fprintf(pOSAStruct->fpOutput, \t\tObject End X: %f\n, pOSAItemInfo->
fObjectEndingHorizontalPosition);
fprintf(pOSAStruct->fpOutput, \t\tObject End Y: %f\n, pOSAItemInfo->
fObjectEndingVerticalPosition);
fprintf(pOSAStruct->fpOutput, \t\tValue Start X: %f\n, pOSAItemInfo->
fValueHorizontalPosition);
fprintf(pOSAStruct->fpOutput, \t\tValue Start Y: %f\n, pOSAItemInfo->
fValueVerticalPosition);
fprintf(pOSAStruct->fpOutput, \t\tValue End X: %f\n, pOSAItemInfo->
fValueEndingHorizontalPosition);
fprintf(pOSAStruct->fpOutput, \t\tValue End Y: %f\n, pOSAItemInfo->
fValueEndingVerticalPosition);
fprintf(pOSAStruct->fpOutput, \t\tValue Text: %s\n, pOSAItemInfo->szValue);
fprintf(pOSAStruct->fpOutput, \t\tFull Object Text: %s\n, pOSAItemInfo->
szFullText);
fprintf(pOSAStruct->fpOutput, \n\t\t****** END ITEM INFO ******\n);
}
else
{
fprintf(pOSAStruct->fpOutput, \n\t\t****** No Item Info At This Point ******
\n);
}
return;
}
/*---------------------------------------------------------------------------
* Function Name: OSA_LinkInfoOut
* Parameters: OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* OSA_LINK_INFO * Pointer to array of Link Info structures
* unsigned long The number of elements in the Link Info array
* Exceptions: None
```

```
* Return Value: None
* Description: Output data from the Link Info structures, if any.
*------------------------------------------------------------------------ */
void OSA_LinkInfoOut(POSASAMPLE_STRUCT pOSAStruct,
POSA_LINK_INFO pOSALinkInfo,
unsigned long ulNumberOfLinks)
{
unsigned short i =0;
/* Check for valid parameter values. If pointers are void, return. */
if (!pOSAStruct || !pOSAStruct->fpOutput)
{
return;
}
/* Check for valid parameter values. If pointer is void or array is empty,
output a message and return. */
if (!pOSALinkInfo || !ulNumberOfLinks)
{
fprintf(pOSAStruct->fpOutput, \n** No Link Information **\n);
return;
}
fprintf(pOSAStruct->fpOutput, \n****** LINK INFO ******\n\n);
for (i=0; i<ulNumberOfLinks; i++)
{
fprintf(pOSAStruct->fpOutput, Lower Left X: %f\n, pOSALinkInfo[i].
fLowerLeftHorizontal);
fprintf(pOSAStruct->fpOutput, Lower Left Y: %f\n, pOSALinkInfo[i].
fLowerLeftVertical);
fprintf(pOSAStruct->fpOutput, Upper Right X: %f\n, pOSALinkInfo[i].
fUpperRightHorizontal);
fprintf(pOSAStruct->fpOutput, Upper Right Y: %f\n, pOSALinkInfo[i].
fUpperRightVertical);
fprintf(pOSAStruct->fpOutput, Application Name: %s\n, pOSALinkInfo[i].
szApplication);
fprintf(pOSAStruct->fpOutput, Form Name: %s\n, pOSALinkInfo[i].szForm);
fprintf(pOSAStruct->fpOutput, Parameter String: %s\n\n, pOSALinkInfo[i].
szParms);
}
fprintf(pOSAStruct->fpOutput, \n****** END LINK INFO ******\n);
return;
}
/*------------------------------------------------------------------------
* Function Name: OSA_FontInfoOut
* Parameters: OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* OSA_FONT_INFO * Pointer to Font Info structure
* Exceptions: None
* Return Value: None
* Description: Output data from the Font Info structure
*------------------------------------------------------------------------ */
void OSA_FontInfoOut(POSASAMPLE_STRUCT pOSAStruct,
POSA_FONT_INFO pFontInfo)
```

```
{
/* Check for valid parameter values. If pointers are void, return. */
if (!pOSAStruct || !pOSAStruct->fpOutput || !pFontInfo)
{
return;
}
fprintf(pOSAStruct->fpOutput, \n****** FONT INFO ******\n\n);
fprintf(pOSAStruct->fpOutput, Font Face Name: %s\n, pFontInfo->lfFaceName);
fprintf(pOSAStruct->fpOutput, Font Point Size: %d\n, pFontInfo->nPointSize);
fprintf(pOSAStruct->fpOutput, Adobe Font Name: %s\n, pFontInfo->
szAdobeFontName);
fprintf(pOSAStruct->fpOutput, \n****** END FONT INFO ******\n);
return;
}
/*------------------------------------------------------------------------
* Function Name: OSA_OpenOutputFile
* Parameters: OSA_REPORT_INFO * Pointer to Report Info Structure
* OSASAMPLE_STRUCT * Pointer to OSA Sample Structure
* Exceptions: None
* Return Value: None
* Description: Create the file which will contain the sample output
*------------------------------------------------------------------------ */
void OSA_OpenOutputFile (POSA_REPORT_INFO pOSAReportInfo,
POSASAMPLE_STRUCT pOSAStruct)
{
/* Formulate the output file name based on information from Report Info. */
strcpy(pOSAReportInfo->szOSAFileName, pOSAReportInfo->szUBEFileName);
#if defined JDENV_AS400
/* On iSeries, the UBE file name is of the form LIBRARY/PRINTQUEUE(F99999),
where 99999 is the job number. We will just switch an O for the F to
indicate an OSA file. */
pStrPtr = strrchr( pOSAReportInfo->szOSAFileName,'F' );
*pStrPtr = 'O';
#else
/* On platforms other than iSeries, just replace the PDF file extension with OSA.⇒
 */
if( !strstr( pOSAReportInfo->szOSAFileName,.pdf ) )
{
/* If there is no .pdf extension, just tack on a .osa extension */
  strcat( pOSAReportInfo->szOSAFileName, .osa );
}
else
{
  sprintf( strstr( pOSAReportInfo->szOSAFileName, .pdf ), .osa);
}
#endif
/* Open the OSA file for output. */
pOSAStruct->fpOutput= fopen(pOSAReportInfo->szOSAFileName, w+b);
if (!pOSAStruct->fpOutput)
{
```

```
/* If the file could not be opened, send an error message back to the UBE
log */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = 1;
}
sprintf(pOSAReportInfo->szLogMessage, Could not open OSA file: %s\n,
pOSAReportInfo->szOSAFileName);
return;
}
return;
}
/*-------------------------------------------------------------------------*/
/* Name: OSASample_StartDoc */
/* Parameters: OSA_REPORT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Open the output file, */
/* Output Report, Section and Object */
/* properties. */
/*-------------------------------------------------------------------------*/
EXTERNC APIEXPORT void CDECL OSASample_StartDoc(OSA_REPORT_INFO*
pOSAReportInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
POSA_SECTION_INFO pOSASectionInfo = NULL;
POSA_OBJECT_INFO pOSAObjectInfo = NULL;
 char *pStrPtr = NULL;
unsigned long i = 0;
unsigned long j = 0;
if(!pOSAReportInfo )
{
return;
}
/* Allocate memory to hold severity value.
Deallocated in OSASample_EndDoc */
if (!pOSAReportInfo->pnLogMessageSeverity)
{
pOSAReportInfo->pnLogMessageSeverity = malloc(sizeof( unsigned short));
}
if (pOSAReportInfo->pnLogMessageSeverity)
{
pOSAReportInfo->pnLogMessageSeverity[0] = 0;
}
/* Create the common structure for passing values between functions,
if it has not been created before this point. */
if (!pOSAReportInfo->pExternalDataPointer)
{
pOSAStruct = malloc(sizeof( OSASAMPLE_STRUCT));
if (pOSAStruct)
```

```
{
memset(pOSAStruct, 0, sizeof(OSASAMPLE_STRUCT));
}
else
{
strcpy(pOSAReportInfo->szLogMessage, OSA: Could not allocate External Data
Pointer.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
}
/* Record the pointer as the external data pointer in Report Info. */
pOSAReportInfo->pExternalDataPointer=pOSAStruct;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at End
Doc.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Create output file if it has not been created yet */
if (!pOSAStruct->fpOutput)
{
OSA_OpenOutputFile (pOSAReportInfo, pOSAStruct);
if (pOSAReportInfo->pnLogMessageSeverity[0] > 0)
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, ****** START DOC EXECUTION POINT ******\n\n);
/* Output Report Info to file. */
OSA_ReportInfoOut(pOSAStruct, pOSAReportInfo);
/* Output Section Info to file. */
if (!pOSAReportInfo->pOSASectionInfo || !pOSAReportInfo->ulNumberOfSections)
{
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = 2;
}
sprintf(pOSAReportInfo->szLogMessage, No Section Info present.\n);
```

```
return;
}
pOSASectionInfo = pOSAReportInfo->pOSASectionInfo;
for (i=0; i<pOSAReportInfo->ulNumberOfSections; i++)
{
OSA_SectionInfoOut(pOSAStruct, pOSASectionInfo);
/* Output Object Info to file. */
if (!pOSASectionInfo->pOSAObjectInfo || !pOSASectionInfo->ulNumberOfObjects)
{
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = 3;
}
sprintf(pOSAReportInfo->szLogMessage,
No Object Info present for Section %s.\n,
pOSASectionInfo->szSectionName);
return;
}
pOSAObjectInfo = pOSASectionInfo->pOSAObjectInfo;
for (j=0; j<pOSASectionInfo->ulNumberOfObjects; j++)
{
OSA_ObjectInfoOut(pOSAStruct, pOSAObjectInfo, 0); /* Do not print Item
Info at this time. */
pOSAObjectInfo++;
}
pOSASectionInfo++;
}
}
/*---------------------------------------------------------------------------- */
/* Name: OSASample_EndDoc */
/* Parameters: OSA_REPORT_INFO*, OSA_PAGEOF_INFO*, unsigned long */
/* Exceptions: None */
/* Return Value: None */
/* Description: Open the output file, */
/* Output Report, Section and Object */
/* properties. */
/*---------------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_EndDoc(OSA_REPORT_INFO*
pOSAReportInfo,
OSA_PAGEOF_INFO* pOSAPageofInfo,
unsigned long ulNumberOfStructs)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely), then
return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
```

```
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at End
Doc.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Close Output File */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** END DOC EXECUTION POINT ******);
if (pOSAStruct->fpOutput)
{
fclose (pOSAStruct->fpOutput);
}
/* Delete the structure created to hold the external data */
free (pOSAStruct);
if (pOSAReportInfo->pnLogMessageSeverity)
free(pOSAReportInfo->pnLogMessageSeverity);
pOSAReportInfo->pExternalDataPointer = NULL;
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_StartPage */
/* Parameters: OSA_REPORT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Report Info to output file. */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_StartPage(OSA_REPORT_INFO*
pOSAReportInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely), then
return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
```

```
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at
Start Page.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at Start Page.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** START PAGE EXECUTION POINT ******\n);
OSA_ReportInfoOut(pOSAStruct, pOSAReportInfo);
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_EndPage */
/* Parameters: OSA_REPORT_INFO*, OSA_LINK_INFO*, unsigned long */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Report Info and Link Info */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_EndPage(POSA_REPORT_INFO
pOSAReportInfo,
POSA_LINK_INFO pOSALinkInfo,
unsigned long ulNumberOfLinks)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely), then
return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
```

```
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at End
Page.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at End
Page.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** END PAGE EXECUTION POINT ******\n);
OSA_ReportInfoOut(pOSAStruct, pOSAReportInfo);
/* Output Link Info */
OSA_LinkInfoOut(pOSAStruct, pOSALinkInfo, ulNumberOfLinks);
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_SetFont */
/* Parameters: OSA_REPORT_INFO*, OSA_FONT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Font Info */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_SetFont(POSA_REPORT_INFO
pOSAReportInfo,
POSA_FONT_INFO pOSAFontInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely),
then return */
if(!pOSAReportInfo )
{
```

```
return;
}
/* Allocate memory to hold severity value.
Deallocated in OSASample_EndDoc */
if (!pOSAReportInfo->pnLogMessageSeverity)
{
pOSAReportInfo->pnLogMessageSeverity = malloc(sizeof( unsigned short));
}
if (pOSAReportInfo->pnLogMessageSeverity)
{
pOSAReportInfo->pnLogMessageSeverity[0] = 0;
}
/* Create the common structure for passing values between functions,
if it has not been created before this point. */
if (!pOSAReportInfo->pExternalDataPointer)
{
pOSAStruct = malloc(sizeof( OSASAMPLE_STRUCT));
if (pOSAStruct)
{
memset(pOSAStruct, 0, sizeof(OSASAMPLE_STRUCT));
}
else
{
strcpy(pOSAReportInfo->szLogMessage, OSA: Could not allocate External
Data Pointer.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
}
/* Record the pointer as the external data pointer in Report Info. */
pOSAReportInfo->pExternalDataPointer=pOSAStruct;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Create output file if it has not been created yet */
if (!pOSAStruct->fpOutput)
{
OSA_OpenOutputFile (pOSAReportInfo, pOSAStruct);
if (pOSAReportInfo->pnLogMessageSeverity[0] > 0)
return;
}
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at Set
Font.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
```

```
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** SET FONT EXECUTION POINT ******\n);
OSA_FontInfoOut(pOSAStruct, pOSAFontInfo);
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_SetColor */
/* Parameters: OSA_REPORT_INFO*, unsigned long int */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Color Reference Number */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_SetColor(POSA_REPORT_INFO
pOSAReportInfo,
unsigned long int zColorRef)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely),
then return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at Set
Color.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at Set
Color.\n);
```

```
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** SET COLOR: %d ******\n, zColorRef);
return;
}
/*------------------------------------------------------------------------ */
/* Name: OSASample_TextOut */
/* Parameters: OSA_REPORT_INFO*, OSA_OBJECT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Font Info */
/* */
/*------------------------------------------------------------------------ */
EXTERNC APIEXPORT void CDECL OSASample_TextOut(POSA_REPORT_INFO
pOSAReportInfo,
POSA_OBJECT_INFO pOSAObjectInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely),
then return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at Text
Out.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at Text
```

```
Out.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** TEXT OUT EXECUTION POINT ******\n);
OSA_ObjectInfoOut(pOSAStruct, pOSAObjectInfo, 1);
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_Underline */
/* Parameters: OSA_REPORT_INFO*, OSA_OBJECT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Font Info */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_DrawUnderline(POSA_REPORT_INFO
pOSAReportInfo,
POSA_OBJECT_INFO pOSAObjectInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely), then return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at
Draw Underline.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
```

```
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at Draw
Underline.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** DRAW UNDERLINE EXECUTION POINT
******\n);
OSA_ObjectInfoOut(pOSAStruct, pOSAObjectInfo, 1);
return;
}
/*---------------------------------------------------------------------- */
/* Name: OSASample_DrawObject */
/* Parameters: OSA_REPORT_INFO*, OSA_OBJECT_INFO* */
/* Exceptions: None */
/* Return Value: None */
/* Description: Output Font Info */
/* */
/*---------------------------------------------------------------------- */
EXTERNC APIEXPORT void CDECL OSASample_DrawObject(POSA_REPORT_INFO
pOSAReportInfo,
POSA_OBJECT_INFO pOSAObjectInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely),
then return */
if(!pOSAReportInfo )
{
return;
}
/* If the external data pointer does not exist execution
cannot go on. Set severity to the highest value, assign a message for the
log and return */
if(!pOSAReportInfo->pExternalDataPointer)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No External Data Pointer at
Draw Object.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAReportInfo->pExternalDataPointer;
```

```
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAReportInfo->szLogMessage, OSA: No Output File pointer at
Draw Object.\n);
/* Set the correct severity to error message severity */
if (pOSAReportInfo->pnLogMessageSeverity)
{
*(pOSAReportInfo->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Identify the Execution Point */
fprintf(pOSAStruct->fpOutput, \n****** DRAW OBJECT EXECUTION POINT ******\n);
OSA_ObjectInfoOut(pOSAStruct, pOSAObjectInfo, 1);
return;
}
```

# Creating and Associating OSA Interfaces

This section provides an overview of OSA interfaces and discusses how to:

- Create OSA interface definitions.
- Associate an OSA interface with an object.

## Understanding OSA Interfaces

Before reports can be output using OSA, you must define the interface. Typically, once defined, OSAs are associated with specific reports or batch versions; however, you can override the default OSA at runtime. You can select from any valid OSA at runtime, although the results might vary depending on how robust the OSA. OSAs can also be associated with environments, hosts, and users or roles, in the same way as default printers.

Depending on the report output requirements, you might need to define multiple interfaces.

### Using the System Hierarchy to Resolve Priority Conflicts

These tables illustrate the hierarchy that the system uses when you associate an OSA interface with more than one object:

| User/Role | Report | Version | Environment | Host |
|---|---|---|---|---|
| username | report | version | environment | hosttype |
| username | report | version | environment | *ALL |
| username | report | version | *ALL | hosttype |
| username | report | version | *ALL | *ALL |
| username | report | *ALL | environment | hosttype |

| User/Role | Report | Version | Environment | Host |
|---|---|---|---|---|
| username | report | *ALL | environment | *ALL |
| username | report | *ALL | *ALL | hosttype |
| username | report | *ALL | *ALL | *ALL |
| username | *ALL | *ALL | environment | hosttype |
| username | *ALL | *ALL | environment | *ALL |
| username | *ALL | *ALL | *ALL | hosttype |
| username | *ALL | *ALL | *ALL | *ALL |

| User/Role | Report | Version | Environment | Host |
|---|---|---|---|---|
| role | report | version | environment | host |
| role | report | version | environment | *ALL |
| role | report | version | *ALL | host |
| role | report | version | *ALL | *ALL |
| role | report | *ALL | environment | host |
| role | report | *ALL | environment | *ALL |
| role | report | *ALL | *ALL | host |
| role | report | *ALL | *ALL | *ALL |
| role | *ALL | *ALL | environment | host |
| role | *ALL | *ALL | environment | *ALL |
| role | *ALL | *ALL | *ALL | host |
| role | *ALL | *ALL | *ALL | *ALL |

| User/Role | Report | Version | Environment | Host |
|---|---|---|---|---|
| *PUBLIC | report | version | environment | host |
| *PUBLIC | report | version | environment | *ALL |
| *PUBLIC | report | version | *ALL | host |
| *PUBLIC | report | version | *ALL | *ALL |
| *PUBLIC | report | *ALL | environment | host |

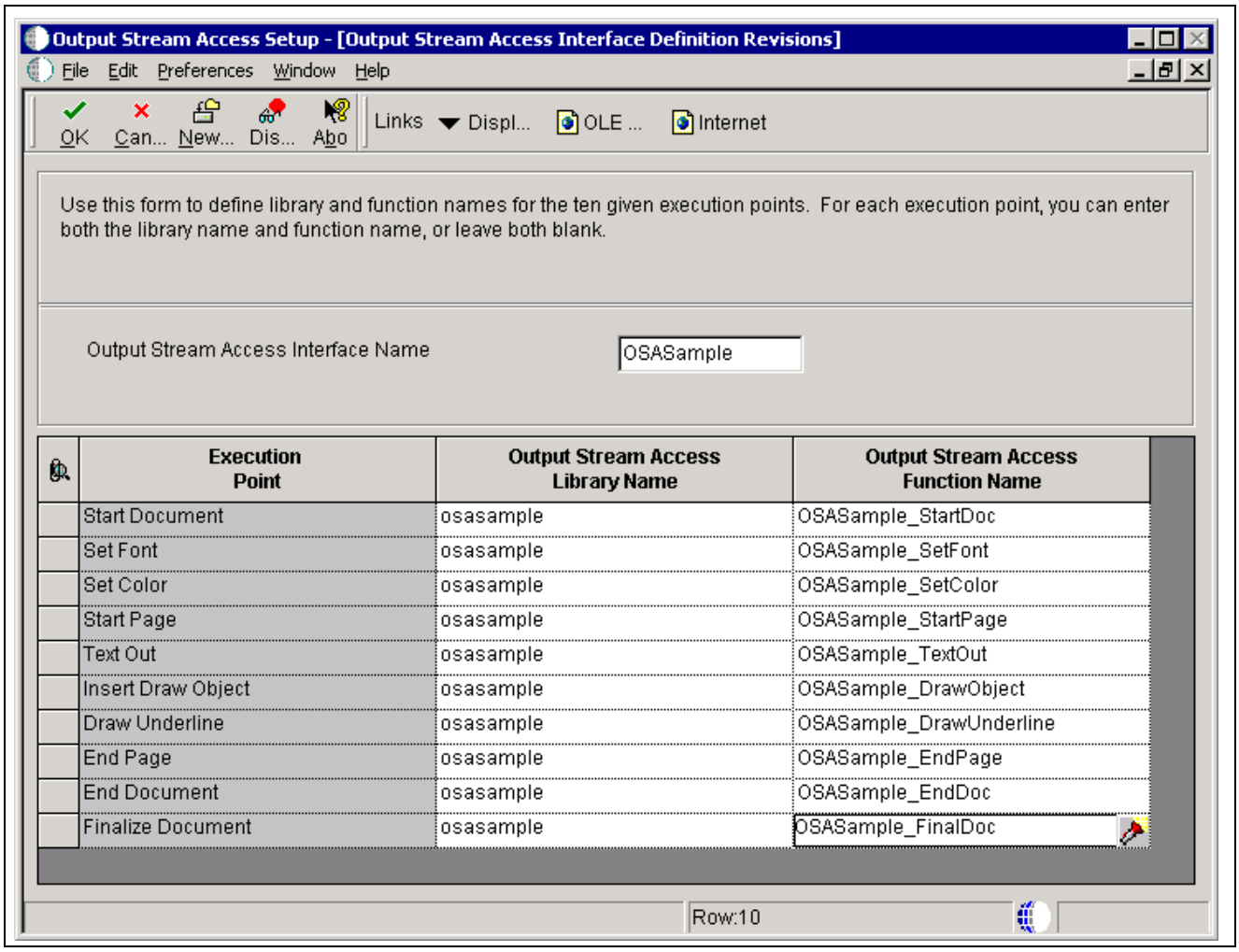
| User/Role | Report | Version | Environment | Host |
|---|---|---|---|---|
| *PUBLIC | report | *ALL | environment | *ALL |
| *PUBLIC | report | *ALL | *ALL | host |
| *PUBLIC | report | *ALL | *ALL | *ALL |
| *PUBLIC | *ALL | *ALL | environment | host |
| *PUBLIC | *ALL | *ALL | environment | *ALL |
| *PUBLIC | *ALL | *ALL | *ALL | *ALL |

## Forms Used to Create and Associate OSA Interfaces

| Form Name | FormID | Navigation | Usage |
|---|---|---|---|
| Output Stream Access Setup | W986168F | EnterpriseOne Life Cycle Tools, Report Management, Batch Processing Setup (GH9013), Output Stream Access Setup | Add or modify OSA interface definitions and add or modify OSA usage specifications. |
| Work With Output Stream Access Interface Definition | W986168C | Click Add or modify the Output Stream Access Interface Definition on the Output Stream Access Setup form. | Add or select an OSA interface definition. |
| Output Stream Access Interface Definition Revisions | W986168I | Click Add on the Work With Output Stream Access Interface Definition form. | Enter the OSA interface name and for each execution point enter the OSA library names and OSA function names. |
| Work With Output Stream Access Interface Usage | W986168D | Click Add or modify the Output Stream Access Interface Usages specification on the Output Stream Access Setup form. | Add or select an OSA interface usage. |
| Output Stream Access Interface Usage Revisions | W986168M | Click Add on the Work With Output Stream Access Interface Usage form. | Enter an OSA interface name and the report, version, environment, host, user or role, and usage status associated with the OSA interface. |

## Creating OSA Interface Definitions

Access the Output Stream Access Interface Definition Revisions form.



Output Stream Access Interface Definition Revisions form

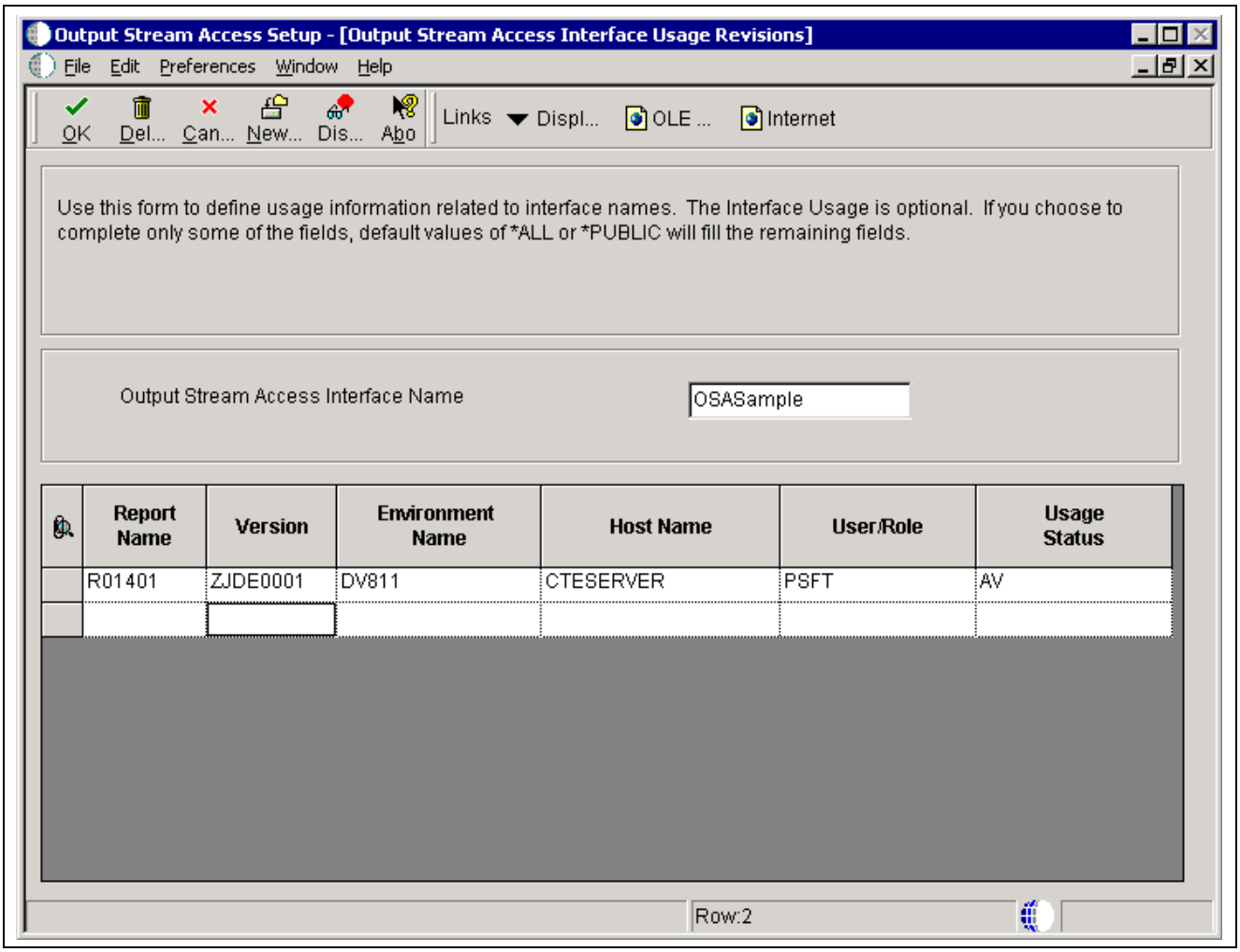
**Output Stream Access Interface Name** A unique name that identifies a set of external functions that can receive and process information during execution. PeopleSoft EnterpriseOne OSA interfaces begin with the letters JDE. It is recommended that you do not begin custom interface names with JDE.

**Output Stream Access Library Name** Shared libraries that contain functions that use the data provided through the OSA interface.

**Output Stream Access Function Name** Functions that conform with the parameters and calling conventions of the OSA interface. The system executes the OSA functions at execution points following a set of parameters. Execution points with no associated function are ignored when the OSA interface executes.

## Associating an OSA Interface with an Object

Access the Batch Processing Setup menu (GH9013).




Output Stream Access Interface Usage Revisions form

**Output Stream Access Interface Name** The OSA interface name to associate with an object. Use the visual assist to select a valid interface name.

**Report Name** The name of the report to associate with the OSA interface. *ALL indicates all reports.

**Version** The name of the batch version to associate with the OSA interface. *ALL indicates all batch versions of the defined report.

**Environment Name** Identifies the location of the report and batch version specifications.

**Host Name** The name of the server that processes the defined batch version.

**User/Role** The user ID or role with permissions to use the OSA interface. *PUBLIC gives permissions to all users.

**Usage Status** Indicates whether the OSA interface is active or not active.

# Glossary of PeopleSoft Terms

**activity** A scheduling entity in PeopleSoft EnterpriseOne Form Design Aid that represents a designated amount of time on a calendar.

**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 PeopleSoft 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** A server in a local area network that contains applications shared by network clients.

**as if processing** A process that enables you to view currency amounts as if they were entered in a currency different from the domestic and foreign currency of the transaction.

**alternate currency** A currency that is different from the domestic currency (when dealing with a domestic-only transaction) or the domestic and foreign currency of a transaction.

In PeopleSoft EnterpriseOne Financial Management, alternate currency processing enables you to enter receipts and payments in a currency other than the one in which they were issued.

**as of processing** A process that is run as of a specific point in time to summarize transactions up to that date. For example, you can run various PeopleSoft EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date.

**back-to-back process** A process in PeopleSoft EnterpriseOne Workflow Management that contains the same keys that are used in another process.

**batch processing** A process of transferring records from a third-party system to PeopleSoft EnterpriseOne.

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

See also direct connect and store-and-forward.

**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-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 view** | A means for selecting specific columns from one or more PeopleSoft EnterpriseOne 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 PeopleSoft 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 PeopleSoft EnterpriseOne that appear on forms in the software. |
| **connector** | Component-based interoperability model that enables third-party applications and PeopleSoft EnterpriseOne to share logic and data. The PeopleSoft EnterpriseOne connector architecture includes Java and COM connectors. |
| **contra/clearing account** | A general ledger account in PeopleSoft EnterpriseOne Financial Management that is used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations in PeopleSoft EnterpriseOne General Accounting. |
| **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. |
| **cost assignment** | The process in PeopleSoft EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects. |
| **cost component** | In PeopleSoft EnterpriseOne Manufacturing Management, an element of an item's cost (for example, material, labor, or overhead). |
| **cross segment edit** | A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced. |
| **currency restatement** | The process of converting amounts from one currency into another currency, generally for reporting purposes. You can use the currency restatement process, for example, when many currencies must be restated into a single currency for consolidated reporting. |
| **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. |

**date pattern** A calendar that represents the beginning date for the fiscal year and the ending date for each period in that year in standard and 52-period accounting.

**denominated-in currency** The company currency in which financial reports are based.

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

**detail information** Information that relates to individual lines in PeopleSoft EnterpriseOne transactions (for example, voucher pay items and sales order detail lines).

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

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

**Do Not Translate (DNT)** A type of data source that must exist on the iSeries because of BLOB restrictions.

**dual pricing** The process of providing prices for goods and services in two currencies.

**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 PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne or PeopleSoft World.

**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 PeopleSoft 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. PeopleSoft 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.

**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.

**facility** An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. A facility is sometimes referred to as a *business unit* .

**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 user 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.

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

**header information** Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows.

**interface table** See Z table.

**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.

**interoperability model** A method for third-party systems to connect to or access PeopleSoft EnterpriseOne.

**in-your-face-error** In PeopleSoft EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.

**IServer service** Developed by PeopleSoft, this internet server service resides on the web server and is used to speed up delivery of the Java class files from the database to the client.

**jargon** An alternative data dictionary item description that PeopleSoft EnterpriseOne or People World displays 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 PeopleSoft 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 PeopleSoft file (or member for iSeries) that provides the runtime settings required for EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne.

**JDENET** PeopleSoft 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 PeopleSoft EnterpriseOne supported platforms.

**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 PeopleSoft EnterpriseOne and World software runs.

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

**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.

**matching document** A document associated with an original document to complete or change a transaction. For example, in PeopleSoft EnterpriseOne Financial Management, a receipt is the matching document of an invoice, and a payment is the matching document of a voucher.

**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 PeopleSoft 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 PeopleSoft 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.

**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.

***nota fiscal*** In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.

***nota fiscal factura*** In Brazil, a nota fiscal with invoice information.

See also *nota fiscal.*

**Object Configuration Manager (OCM)** In PeopleSoft 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-in capabilities for developers, and it controls the creation, modification, and use of PeopleSoft 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 PeopleSoft EnterpriseOne data for summarization and report generation.

**Output Stream Access (OSA)** An interoperability model that enables you to set up an interface for PeopleSoft EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.

**package** 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 snap shot 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 PeopleSoft 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.

**PeopleSoft Database** See JDEBASE Database Middleware.

**planning family** A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate.

**preference profile** The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups.

**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 PeopleSoft EnterpriseOne environment used to test unaltered objects with PeopleSoft 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 PeopleSoft EnterpriseOne environment in which users operate EnterpriseOne software.

**production-grade file server** A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services.

**program temporary fix (PTF)** A representation of changes to PeopleSoft software that your organization receives on magnetic tapes or disks.

**project** In PeopleSoft 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 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.

**pull replication** One of the PeopleSoft methods for replicating data to individual workstations. Such machines are set up as pull subscribers using PeopleSoft EnterpriseOne data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the F98DRPCN table.

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

**real-time event** A service that uses system calls to capture PeopleSoft EnterpriseOne transactions as they occur and to provide notification to third-party software, end users, and other PeopleSoft systems that have requested notification when certain transactions occur.

**refresh** A function used to modify PeopleSoft 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, such as B73.2 or B73.2.1.

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

**quote order** In PeopleSoft EnterpriseOne Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order.

| | |
|---|---|
| | In PeopleSoft EnterpriseOne Sales Order Management, item and price information for a customer who has not yet committed to a sales order. |
| **selection** | Found on PeopleSoft 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. |
| **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. It also updates the Server Plan detail record to reflect completion. |
| **spot rate** | An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies. |
| **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 PeopleSoft 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. |
| **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. |
| **supplemental data** | Any type of information that is not maintained in a master file. Supplemental data is usually additional information about employees, applicants, requisitions, and jobs (such as an employee's job skills, degrees, or foreign languages spoken). You can track virtually any type of information that your organization needs.<br><br>For example, in addition to the data in the standard master tables (the Address Book Master, Customer Master, and Supplier Master tables), you can maintain other kinds of data in separate, generic databases. These generic databases enable a standard approach to entering and maintaining supplemental data across PeopleSoft EnterpriseOne systems. |
| **table access management (TAM)** | The PeopleSoft 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 PeopleSoft EnterpriseOne and third-party systems using non-PeopleSoft EnterpriseOne tables. |
| **table conversion** | An interoperability model that enables the exchange of information between PeopleSoft EnterpriseOne and third-party systems using non-PeopleSoft 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 PeopleSoft 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. |

**three-tier processing** The task of entering, reviewing and approving, and posting batches of transactions in PeopleSoft EnterpriseOne.

**three-way voucher match** In PeopleSoft EnterpriseOne Procurement and Subcontract Management, the process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records to create vouchers.

**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 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.

**two-way voucher match** In PeopleSoft EnterpriseOne Procurement and Subcontract Management, the process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information.

**User Overrides merge** Adds new user override records into a customer's user override table.

**variance** In Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.

In EnterpriseOne Project Costing and EnterpriseOne Manufacturing Management, the difference between two methods of costing the same item (for example, the difference between the frozen standard cost and the current cost is an engineering variance). Frozen standard costs come from the Cost Components table, and the current costs are calculated using the current bill of material, routing, and overhead rates.

**Version 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 PeopleSoft EnterpriseOne or World form or report.

**wchar_t** An internal type of a wide character. It is used for writing portable programs for international markets.

**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.

**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.

**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 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 PeopleSoft EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench.

**work day calendar** In EnterpriseOne Manufacturing Management, a calendar that is used in planning functions that consecutively lists only working days so that component and work order scheduling can be done based on the actual number of work days available. A work day calendar is sometimes referred to as planning calendar, manufacturing calendar, or shop floor calendar.

**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 PeopleSoft EnterpriseOne transactions as they occur and then calls third-party software, end users, and other PeopleSoft 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 PeopleSoft EnterpriseOne for responses.

**XML List** An interoperability capability that enables you to request and receive PeopleSoft EnterpriseOne database information in chunks.

**XML Service** An interoperability capability that enables you to request events from one PeopleSoft EnterpriseOne system and receive a response from another PeopleSoft EnterpriseOne system.

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

**XML Transaction Service (XTS)** Transforms an XML document that is not in the PeopleSoft EnterpriseOne format into an XML document that can be processed by PeopleSoft EnterpriseOne. XTS then transforms the response back to the request originator XML format.

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

**Z table** A working table where non-PeopleSoft EnterpriseOne information can be stored and then processed into PeopleSoft EnterpriseOne. Z tables also can be used to retrieve PeopleSoft 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 PeopleSoft EnterpriseOne database.

# Index


**A**

additional documentation viii
Advanced Conversion Program form 27
application fundamentals vii

**B**

Bar Code Support Revisions form 29
barcode fonts
  copying information for new printers 30
  deleting support information from printers 31
  modifying printer information 30
  setting up 29
  understanding 29
barcode support information, deleting from printers 31
batch jobs
  submitting on Microsoft Windows client 12
  submitting using report interconnects 13
batch versions
  designing to print on line printers 32
  locating PDFs when run on the Microsoft Windows client 35
  locating PDFs when run on the web client 35
  printing 17
  running locally on the Microsoft Windows client 35
  running on the server 35
  submitting using report interconnects 13
  understanding submission of 34

**C**

code page, order of precedence for PCL printing 32
comma separated value files, *See* CSV
comments, submitting xii
common elements xiii
contact information xii
conversions, *See* null pass-through print filters
cross-references xi
CSV
  defining at runtime 7, 14
  defining in batch versions 7
  defining in Report Design Aid 9
  defining in report templates 6
  design recommendations 7
  exporting to 6
  files created when exporting to 7
Customer Connection website viii

**D**

Default Printer Revisions form 25
default printers
  defining 25
  understanding 20
device context, using to create PDF files 34
documentation
  printed viii
  related viii
  updates viii
draw underline parameters, defining 43

**E**

end document parameters, defining 43
end page parameters, defining 43

**F**

finalize document parameters, defining 43
function parameters
  defining draw underline parameters 43
  defining end document parameters 43
  defining end page parameters 43
  defining finalize document parameters 43
  defining insert draw object parameters 42
  defining set color parameters 42
  defining set font parameters 42
  defining start document parameters 42
  defining start page parameters 42
  defining text out parameters 42
  understanding 41
function signatures, understanding 41

**H**

hierarchy
  print properties 5
  printers 6

**I**

IFS
  defining the PrintQueue directory 13
  working with 13
include files 43
initialization files
  jas.ini 36
  jde.ini 36
  modifying print settings 36
insert draw object parameters, defining 42
integrated file system, *See* IFS
iSeries
  adding printers 18
  defining to print multiple copies of reports to a remote printer 34
  understanding printing multiple copies to a remote 33
  using IFS 13

**J**

jas.ini
  defining the print immediate option 14
  modifying PrintImmediate and KeepUBE settings 36
jde.ini
  defining the print immediate option 14
  defining the SavePDL file option 15
  modifying PrintImmediate and SaveOutput settings 36
JDEOSA.H file 43

**K**

K2DoInitPrinter 8
KeepUBE, defining in the jas.ini 36

**L**

line printers
  defining to print multiple copies of reports on remote iSeries 34
  designing reports to print on line printers 32
  modifying reports to print on 33
  understanding printing multiple copies on remote iSeries 33
logging, activating at runtime 35

**M**

Microsoft Windows client
  locating log files 35
  locating PDF files 35
  running batch versions locally 35
MMA Partners viii
multiple code sets
  understanding for PCL 31

**N**

notes xi
null pass-through print filters
  adding 27
  understanding 20

**O**

order of precedence, for PCL printing 32
orientation, specifying at runtime 14
OSA
  associating interfaces with objects 70
  benefits of using 8
  creating interface definitions 69
  naming and locating files 47
  resolving priority conflicts 67
  retrieving documents 43
  understanding 8, 39
  understanding function parameters 41
  understanding function signatures 41
  understanding interfaces 67
  understanding libraries 41
OSA documents, retrieving 43
OSA file names 48
OSA interfaces
  associating with objects 70
  creating definitions 69
  understanding 67
OSA Libraries
  naming and locating files 47
  understanding 41
OSASample source code
  components 48
  OSASample.c example 49
  OSASample.h example 48
  OSAStruct.h 48
OSASample.c, source code example 49
OSASample.h, source code example 48
OSAStruct.h, source code example 48

output
defining in jde.ini and jas.ini 36
defining PrintQueue directory 13
submitting batch jobs locally on the Microsoft Windows client 12
understanding 3
output management, understanding 3
Output Stream Access, *See* OSA

## P

paper
selecting types 6
selecting types at runtime 13
paper types
defining in Report Design Aid 8
deleting 27
selecting at runtime 13
understanding 6, 19
PDF
locating when batch versions are run on the Microsoft Windows client 35
locating when batch versions are run on the web client 35
PDL
defining the SavePDL option 15
understanding 19
PeopleBooks
ordering viii
PeopleCode, typographical conventions x
PeopleSoft application fundamentals vii
Platform Information form 21
platform information, defining 18
prerequisites vii
print filters
adding null pass-through filters 27
understanding null pass-through filters 20
print immediate option, defining 14
print orientation, specifying at runtime 14
print properties
hierarchy 5
modifying 5
understanding 5
print settings, understanding the order of precedence for PCL printing 32
Print Setup form 8
printed documentation viii
Printer Definition Language, *See* PDL
printer information
copying barcode information for new printers 30
modifying for barcode fonts 30
storing and passing 12
printer records, searching for incorrect records 28
Printer Search & Select form 8
Printer Setup form 22
printers
adding for iSeries 18
adding for UNIX 18
adding for Windows NT 18
copying 27
copying barcode information for new printers 30
defining default printers 25
defining in Report Design Aid 8
defining paper types 19
defining platform information 18
defining the Printer Definition Language (PDL) 19
defining the PrintQueue directory for the IFS 13
deleting 27
deleting barcode support information 31
deleting paper types 27
designing reports to print on line printers 32
determining based on hierarchical structure 6, 13
exporting to CSV at runtime 14
modifying 27
modifying barcode information 30
modifying reports to print on line printers 33
modifying settings in initialization files 36
overriding designated printers 5
overriding print-time characteristics 36
printing multiple copies to remote iSeries line printers 34
resolving 13
searching for incorrect printer records 28
specifying print orientation at runtime 14
storing and passing printer information 12
understanding defining defaults 20

understanding print properties at runtime 11
understanding printing multiple copies on remote iSeries 33
using the iSeries 13
Printers application, understanding 18
Printers form 21
PrintImmediate
defining in the jas.ini 36
defining in the jde.ini 36
printing administration, understanding 17
PrintQueue
defining for IFS 13
locating PDF files 35
priority conflicts, using the system hierarchy to resolve 67

## R

related documentation viii
remote printers
printing multiple copies on the iSeries 34
understanding printing multiple copies on iSeries 33
Report Design Aid
defining batch applications to export to CSV 6
defining batch versions to export to CSV 7
report output, understanding 3
reports
designing to print on line printers 32
printing 17
runtime
activating logging 35
defining print properties 5
determining printer based on hierarchical structure 13
exporting to CSV 7, 14
overriding OSA interface 67
overriding printer 6
reading the jde.ini and jas.ini 4
selecting paper types 13
selecting print orientation 14

## S

SaveOutput, defining in the jde.ini 36
SavePDL file option, defining 15
servers, running batch versions 35
set color parameters, defining 42
set font parameters, defining 42
start document parameters, defining 42
start page parameters, defining 42
suggestions, submitting xii
symbol set, order of precedence for PCL printing 32
system functions
initializing logical printer name 8
using K2DoInitPrinter 8
system hierarchy
using to resolve priority conflicts 67

## T

temp directory, locating PDF files run on the web client 35
text out parameters, defining 42
typographical conventions x

## U

UNIX, adding printers 18

## V

visual cues xi

## W

warnings xi
web client
locating log files 35
locating PDF files 35
modifying the jas.ini 36
Windows NT, adding printers 18
Work With Bar Code Font form 30
Work With Batch Versions - Available Versions form 28