

**JD Edwards EnterpriseOne Tools**  
Report Printing Administration Technologies Guide  
Release 9.1.x  
**E24250-02**

April 2013

Copyright © 2011, 2013, Oracle and/or its affiliates. All rights reserved.

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

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

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

**U.S. GOVERNMENT END USERS:** Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

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

---

---

# Contents

<b>Preface .....</b>	vii
Audience.....	vii
Documentation Accessibility .....	vii
.....	vii
Related Documents .....	vii
Conventions .....	viii
<b>1 Introduction to JD Edwards EnterpriseOne Report Printing Administration Technologies</b>	
1.1 JD Edwards EnterpriseOne Report Printing Administration Technologies Overview....	1-1
1.2 JD Edwards EnterpriseOne Report Printing Administration Technologies Implementation.....	1-1
<b>2 Understanding Report Printing Administration Technologies</b>	
2.1 Report Output .....	2-1
2.2 Output Management.....	2-2
<b>3 Defining Print Properties for Reports</b>	
3.1 Understanding Print Properties.....	3-1
3.2 Modifying Print Properties.....	3-1
3.2.1 Understanding Designated Printers .....	3-1
3.2.2 Understanding the Initialize Logical Printer Name System Function.....	3-2
3.2.3 Understanding Paper Types .....	3-2
3.2.4 Understanding Exporting to CSV .....	3-3
3.2.5 Understanding OSA Interfaces.....	3-5
3.2.6 Setting Print Properties in RDA .....	3-5
3.2.6.1 Prerequisite.....	3-5
3.2.6.2 Defining Printers.....	3-5
3.2.6.3 Selecting Paper Types .....	3-5
3.2.6.4 Exporting to CSV .....	3-6
<b>4 Understanding Print Properties at Runtime</b>	
4.1 Batch Version Submission .....	4-1
4.2 Jobs Submitted from the Microsoft Windows Client.....	4-2

4.3	Printer Information.....	4-2
4.4	Customizing the Location of Report Output .....	4-3
4.5	Printer Selections at Runtime .....	4-3
4.6	Paper Type Selections at Runtime .....	4-4
4.7	Print Orientation Selections at Runtime .....	4-4
4.8	Export to CSV at Runtime.....	4-4
4.9	Print Settings in the Initialization Files.....	4-5
4.9.1	The Print Immediate Setting .....	4-5
4.9.1.1	Print Immediate--Microsoft Windows Client.....	4-5
4.9.1.2	Print Immediate--HTML Client.....	4-5
4.9.1.3	Print Immediate--Interconnected Reports .....	4-6
4.9.1.4	Print Immediate--Microsoft Windows Server .....	4-6
4.9.2	The SavePDL Setting .....	4-6

## 5 Working with Report Printing Administration

5.1	Understanding Report Printing Administration.....	5-1
5.2	Working with the JD Edwards EnterpriseOne Printer Application.....	5-2
5.2.1	Understanding the JD Edwards EnterpriseOne Printer Application .....	5-2
5.2.1.1	Platform Information .....	5-2
5.2.1.2	Printer Definition Language .....	5-3
5.2.1.3	Paper Types .....	5-4
5.2.1.4	Default Printers.....	5-4
5.2.2	Understanding Pass-Through Print Filters.....	5-5
5.2.3	Forms Used to Add Printers .....	5-5
5.2.4	Adding Printers.....	5-6
5.2.4.1	Platform Information .....	5-6
5.2.4.2	Printer Setup.....	5-7
5.2.4.3	Printer Setup.....	5-9
5.2.5	Defining Default Printers.....	5-11
5.2.6	Modifying Printers .....	5-12
5.2.7	Copying Printers .....	5-12
5.2.8	Deleting Printers .....	5-12
5.2.9	Deleting Paper Types .....	5-12
5.2.10	Selecting a Print Style Option (Release 9.1 Update 3) .....	5-13
5.2.11	Adding Pass-through Print Filters .....	5-13
5.2.12	Searching for Incorrect Printer Records .....	5-13
5.3	Setting Up Barcode Fonts.....	5-14
5.3.1	Understanding Barcode Fonts .....	5-14
5.3.2	Forms Used to Set Up Printers to Use Barcode Fonts .....	5-14
5.3.3	Setting Up Printers to Use Barcode Fonts .....	5-14
5.3.3.1	PCL Printer .....	5-15
5.3.3.2	PostScript Printer.....	5-15
5.3.3.3	Printer Configuration Troubleshooting .....	5-16
5.3.4	Modifying Barcode Printer Information.....	5-16
5.3.5	Copying Barcode Printer Information for New Printers .....	5-16
5.3.6	Deleting Barcode Support Information from Printers.....	5-16
5.4	Understanding Multiple Code Sets for PCL .....	5-16

5.4.1	Multiple Code Sets for PCL Printing .....	5-17
5.4.2	The Order of Precedence for PCL Printing .....	5-17
5.4.2.1	Code Page Order of Precedence .....	5-17
5.4.2.2	Symbol Set Order of Precedence .....	5-18
5.5	Designing Reports to Print on Line Printers .....	5-18
5.5.1	Understanding Reports Designed to Print on Line Printers .....	5-18
5.5.2	Modifying Reports to Print on Line Printers .....	5-19
5.5.2.1	Object Alignment.....	5-19
5.5.2.2	Group Sections.....	5-19
5.5.3	Defining the Printer for Line Printing .....	5-20
5.5.3.1	Line Printing.....	5-20
5.5.3.2	Epson and IBM Line Printer Settings .....	5-20
5.6	Printing Reports .....	5-21
5.6.1	Batch Versions at Submission .....	5-21
5.6.2	Batch Versions Processed on the Server .....	5-22
5.6.3	Batch Versions Processed Locally from the Microsoft Windows Client .....	5-22
5.6.4	Print-Time Characteristics.....	5-23
5.6.5	Print Settings for Batch Versions .....	5-23

## 6 Working with Output Stream Access

6.1	Understanding OSA .....	6-1
6.2	Creating OSA Libraries.....	6-3
6.2.1	OSA Libraries .....	6-3
6.2.1.1	Function Signatures.....	6-3
6.2.2	Function Parameters.....	6-3
6.2.2.1	Defining Start Document Parameters.....	6-4
6.2.2.2	Defining Set Font Parameters .....	6-4
6.2.2.3	Defining Set Color Parameters .....	6-4
6.2.2.4	Defining Start Page Parameters.....	6-4
6.2.2.5	Defining Text Out Parameters .....	6-4
6.2.2.6	Defining Insert Draw Object Parameters .....	6-4
6.2.2.7	Defining Draw Underline Parameters.....	6-5
6.2.2.8	Defining End Page Parameters.....	6-5
6.2.2.9	Defining End Document Parameters.....	6-5
6.2.2.10	Defining Finalize Document Parameters .....	6-5
6.2.3	OSA Documents.....	6-5
6.2.4	Include Files.....	6-5
6.2.5	File Locations and Names .....	6-5
6.2.6	OSASample Source Code .....	6-6
6.2.6.1	OSAStruct.h.....	6-6
6.2.6.2	OSASample.h .....	6-6
6.2.6.3	OSASAMPLE.c.....	6-7
6.3	Creating and Associating OSA Interfaces .....	6-22
6.3.1	Understanding OSA Interfaces.....	6-22
6.3.1.1	Using the System Hierarchy to Resolve Priority Conflicts.....	6-22
6.3.2	Forms Used to Create and Associate OSA Interfaces.....	6-23
6.3.3	Creating OSA Interface Definitions .....	6-23

6.3.4	Associating an OSA Interface with an Object.....	6-24
-------	--	------

## **Glossary**

## **Index**

---

---

# Preface

Welcome to the JD Edwards EnterpriseOne Tools Report Printing Administration Technologies Guide

---

**Note:** This guide has been updated for JD Edwards EnterpriseOne Tools Release 9.1 Update 3. For details on documentation updates, refer to the *JD Edwards EnterpriseOne Tools Net Change Guide*.

---

## Audience

This guide is intended for developers and administrators who manage report output. Report output management refers to the managing of the different output options that are available for viewing a report.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## Related Documents

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

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

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

This guide contains references to server configuration settings that JD Edwards EnterpriseOne stores in configuration files (such as jde.ini, jas.ini, jdbj.ini, jdelog.properties, and so on). Beginning with the JD Edwards EnterpriseOne Tools Release 8.97, it is highly recommended that you only access and manage these settings for the supported server types using the Server Manager program.

See the *JD Edwards EnterpriseOne Server Manager Guide*.

## Conventions

The following text conventions are used in this document:

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

---

# Introduction to JD Edwards EnterpriseOne Report Printing Administration Technologies

This chapter contains the following topics:

- [Section 1.1, "JD Edwards EnterpriseOne Report Printing Administration Technologies Overview"](#)
- [Section 1.2, "JD Edwards EnterpriseOne Report Printing Administration Technologies Implementation"](#)

## 1.1 JD Edwards EnterpriseOne Report Printing Administration Technologies Overview

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

## 1.2 JD Edwards EnterpriseOne Report Printing Administration Technologies Implementation

This section provides an overview of the steps that are required to implement JD Edwards EnterpriseOne Report Printing Administration Technologies.

In the planning phase of the implementation, take advantage of all JD Edwards sources of information, including the installation guides and troubleshooting information.

The following list contains the high-level steps for the JD Edwards EnterpriseOne Report Printing Administration Technologies implementation.

- Set up permissions to access and use JD Edwards EnterpriseOne Object Management Workbench (OMW) and the JD Edwards EnterpriseOne Printer Application (P98616) using JD Edwards EnterpriseOne Security Workbench.  
See "Managing Application Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.
- Add yourself to the system in a developer role so that you have permissions to create and modify JD Edwards EnterpriseOne objects.  
See "Setting Up User Roles" in the *JD Edwards EnterpriseOne Tools Object Management Workbench Guide*.
- Set up permissions to create OMW projects.

See "Setting Up Allowed User Actions" in the *JD Edwards EnterpriseOne Tools Object Management Workbench Guide*.

- Set up save locations to enable you to save JD Edwards EnterpriseOne objects that are not ready to be checked in.

See "Configuring Object Save Locations" in the *JD Edwards EnterpriseOne Tools Object Management Workbench Guide*.

---

# Understanding Report Printing Administration Technologies

This chapter contains the following topics:

- [Section 2.1, "Report Output"](#)
- [Section 2.2, "Output Management"](#)

## 2.1 Report Output

In JD Edwards EnterpriseOne Report Design Aid (RDA), you create reports and define specific printing properties to affect the report output. The report developer creates 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 these formats:

- PDF  
View the report in Adobe Reader.
- Comma separated values (CSV)  
Export the report to a CSV spreadsheet application.
- Output stream access (OSA)  
Export the report to a third-party product using OSA interfaces.
- Printed copy  
Send the report to a printer.
- BI Publisher Report  
The BI Publisher report can be a PDF, RTF, XLS, Power Point, etext, XML, and so on.

See "Creating BI Publisher Report Definitions" in the *JD Edwards EnterpriseOne Tools BI Publisher for JD Edwards EnterpriseOne Guide*.

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 these formats:

- PDF
- CSV
- OSA
- Printed copy
- BI Publisher Report

The BI Publisher report can be a PDF, RTF, XLS, Power Point, etext, XML, and so on.

See "Creating BI Publisher Report Definitions" in the *JD Edwards EnterpriseOne Tools BI Publisher for JD Edwards EnterpriseOne Guide*.

Viewing the report on screen is not an option at runtime from the web client. However, once the report has been processed, you can view the report online using options on the Row menu.

## 2.2 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 types. JD Edwards 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 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 HTML (JAS) server.

---

## Defining Print Properties for Reports

This chapter contains the following topics:

- [Section 3.1, "Understanding Print Properties"](#)
- [Section 3.2, "Modifying Print Properties"](#)

### 3.1 Understanding Print Properties

You add a new printer to the system and define the print properties using the JD Edwards EnterpriseOne Printers (P98616) application. You define printers for use by a specific user ID, user role, or for all users. You have the option of associating printers with:

- A specific report.
- A specific version of a report.
- All reports.
- JD Edwards EnterpriseOne environments

JD Edwards EnterpriseOne reporting incorporates printing properties to determine the format of the report output.

### 3.2 Modifying Print Properties

This section contains the following topics:

- Understanding Designated Printers.
- Understanding the Initialize Logical Printer Name System Function.
- Understanding Paper Types.
- Understanding Exporting to CSV.
- Understanding OSA Interfaces.
- Setting Print Properties in RDA.

#### 3.2.1 Understanding Designated Printers

The system administrator defines the default printer to be used with all batch processes. The printer is associated with a user ID, user role or with \*PUBLIC, a default value that includes all users. The printer also can be associated with a specific JD Edwards EnterpriseOne environment. 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 user ID overrides a printer associated with a specific user role. A printer associated with a specific user role overrides a printer associated with \*PUBLIC and a specific batch process. You can override these printer associations by selecting a printer using Print Setup on the File menu in RDA. The printer that you select in RDA is stored in the print specifications, causing the report to always print to the printer that you defined unless overridden at runtime.

When defining a printer in RDA, consider the hierarchy that the system uses to determine the printer that is used at runtime. Assuming there is no user-defined override at runtime, the system looks for the printer defined in RDA, if no printer is defined, the system uses the default printer defined in the JD Edwards EnterpriseOne Printer Application (P98616.)

### 3.2.2 Understanding the Initialize Logical Printer Name System Function

You can use the *Do Initialize Printer* event to specify a printer to be used by the system 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 settings 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 jddebug log.

### 3.2.3 Understanding Paper Types

When defining the paper type for reports, you can select from predefined paper sizes or you can enter custom paper dimensions. The standard predefined selections available in P98616 are:

- A4
- Legal
- Letter

You must define one of these selections as the default paper type. You can override the default paper type from Print Setup in RDA.

The paper types defined in P98616 are stored in the Paper Definition (F986162) table. RDA inherits the paper size from this table. Additional paper types are defined for use in RDA using P98616. This table shows some examples of additional paper types:

Paper Type	Size (uom)
Tabloid	11 x 17 in
A3	297 x 420 mm

Paper Type	Size (uom)
B4 (JIS)	250 x 354 mm

You can use a selection of different units of measurement to define custom paper sizes in P98616. The minimum definable *width* in inches is two inches, and the maximum is 21 inches. The minimum definable *height* in inches is two inches and the maximum is 24 inches. In RDA, a report developer can define custom paper sizes from the Print Setup form.

See [Selecting Paper Types](#)

### 3.2.4 Understanding Exporting to CSV

CSV files are used to output tabular data. In addition to viewing the report in a CSV file, you can manipulate the report data after the report finishes processing. To view report data in a spreadsheet program, such as Microsoft Excel or Lotus 123, select the export to a CSV option. You can select the CSV option using these methods, each with a different result:

- In RDA 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 versions are run.

Select a report template and in RDA, select the *Export to CSV* option in Print Setup.

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

- In RDA for the batch version.

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

Select a batch version and in RDA, 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. When a batch version is defined to export to CSV for every instance, you can clear the *Export to CSV* option at runtime if 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 for a *single* submission only.

When running batch versions locally, select the *Export to CSV* option to submit the batch version.

When running batch versions 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 and follow these recommendations:

Recommendation	Description
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 RDA. For best results, use this 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 displays in discrete columns. You can wrap the text in a cell after the data is exported to the spreadsheet. Delete unused columns in the spreadsheet and reformat information as needed.
Align data fields vertically.	If data fields are not aligned vertically, they display in separate rows in the spreadsheet. 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 process occupies the cell in the spreadsheet.
Format 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.	<p>In these countries, the decimal separator is recognized as a comma when the report exports. Tabs are stripped out instead of commas and a tab-separated file with a .txt extension is created.</p> <p>The information transfers as flat text, so totaling columns display only text. You must then set up totaling in the spreadsheet program.</p>

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 is displayed by a spreadsheet program such as Microsoft Excel or Lotus 123, which launches automatically when you run the batch version locally.

When you run the batch version on a server, select View CSV from the Submitted Job Search form 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.

The Export to CSV option recognizes when the decimal separator is a comma, and rather than creating a comma delimited CSV file, it creates a tab-separated file with a .txt extension that can be opened in Notepad. To create a CSV file when the comma is a decimal separator, make the following changes in the jde.ini.

```
[UBE]
prtCSVExtension=.csv
PRTCSVSeparator=,
```

Other file extensions and separators can be used by changing the jde.ini settings.

Reports that are processed on the enterprise server use the settings in the jde.ini of the enterprise server rather than the workstation jde.ini. Therefore, when making the above jde.ini setting changes, they need to be made on both the workstation and enterprise server.

### 3.2.5 Understanding OSA Interfaces

You can select to output reports to third-party software programs using OSA. OSA interfaces enable the third-party program to process and format the data concurrently with the processing of a UBE.

See [Understanding OSA](#)

### 3.2.6 Setting Print Properties in RDA

When you design a report in RDA, you can override the default printer settings configured in the Printer Application (P98616) to print the report. To print a report from RDA, you can do the following:

- Define printers
- Select paper types
- Export to CSV

#### 3.2.6.1 Prerequisite

Before defining printers in RDA, check out an existing report template.

See "Understanding How to Open Existing Reports" in the *JD Edwards EnterpriseOne Tools Report Design Aid Guide*.

#### 3.2.6.2 Defining Printers

From EnterpriseOne Life Cycle Tools menu, 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 then click **Select**.

#### 3.2.6.3 Selecting Paper Types

A report developer can define a custom paper type in RDA. The page type setting in RDA is a visual guide only and has no effect on the page type of the report that you submit unless you select the *Custom* option in the Printer Setup form. If you do not select the *Custom* option, your report processes using the default printer setup. When you select the *Custom* option, the custom definitions that you set up in RDA override the definitions that are set up in P98616. Selecting the *Custom* option ensures the report always processes using the custom page type. If you do not select the *Custom* option, the page type processes using the default page type in the report specifications entered in P98616, even though you defined a different page type.

In RDA, select Print Setup from the File menu to 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.

#### **3.2.6.4 Exporting to CSV**

Before you export a report to CSV from RDA, verify that no columns or fields in the report overlap. Move any overlapping columns or fields.

In RDA, select **Print Setup** from the **File** menu to access the **Print Setup** form.

1. Under **OneWorld Printer**, select **Export to CSV**, and then click **OK**.
2. If prompted to automatically set the grid size, click **OK**; otherwise, do the following:
  - From the **Layout** menu, select **Grid Alignment**.
  - On the **Alignment Grid** form, set the horizontal spacing to **52**.
3. Select the **Snap to Grid** option, and then click **OK**.

The system applies these settings to the entire report.

---

# Understanding Print Properties at Runtime

This chapter contains the following topics:

- [Section 4.1, "Batch Version Submission"](#)
- [Section 4.2, "Jobs Submitted from the Microsoft Windows Client"](#)
- [Section 4.3, "Printer Information"](#)
- [Section 4.4, "Customizing the Location of Report Output"](#)
- [Section 4.5, "Printer Selections at Runtime"](#)
- [Section 4.6, "Paper Type Selections at Runtime"](#)
- [Section 4.7, "Print Orientation Selections at Runtime"](#)
- [Section 4.8, "Export to CSV at Runtime"](#)
- [Section 4.9, "Print Settings in the Initialization Files"](#)

## 4.1 Batch Version Submission

You must use a batch version to process a report. You can submit a batch version to run locally on a Microsoft Windows client, or you can use any client to submit the batch version to a server.

When you submit a batch version locally, the Microsoft Windows client immediately launches the batch engine process. You have the option to view the output on screen or send it to a printer. This method is used primarily by UBE developers.

The server handles batch processing more efficiently than the workstation. When 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 batch versions to the server using these methods:

- The JD Edwards EnterpriseOne Batch Versions (P98305) application on the Microsoft Windows client.
- Another report or interactive application using a report interconnect.

You must define the report interconnect in JD Edwards EnterpriseOne Report Design Aid (RDA). 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 JD Edwards EnterpriseOne Batch Versions - Web Version (P98305W) application on the web client.

On the web client, batch versions are submitted *only* to the server.

- From the JD Edwards EnterpriseOne Menu on the web client.  
For easy access, you can add a report as a task on the JD Edwards EnterpriseOne Menu.
- From the command line on the server.  
See "Submitting Batch Versions" in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

## 4.2 Jobs Submitted from the Microsoft Windows Client

On the Microsoft Windows client, you can select from these options when submitting batch jobs:

Output Option	Description
On Screen.	Enables you to generate a PDF file that is displayed through Adobe Reader. Adobe Reader is launched by the system when the processing of the batch version is complete.
To Printer.	Enables you to modify output options from the Printer Selection form.
Export to CSV.	Enables you to generate both a CSV and a PDF file. The report is displayed through a CSV viewer, such as Microsoft Excel. The CSV viewer is launched automatically by the system when processing of the batch version is complete.
Export using output stream access (OSA).	Enables you to export the report to a third-party software application. The location of the output is determined by the OSA interface. For example, the JD Edwards 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 "Submitting Batch Versions" in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

---

**Note:** For advanced version prompting options that apply to BI Publisher for JD Edwards EnterpriseOne functionality, see "Accessing Batch Version Advanced Option Overrides" in the *JD Edwards EnterpriseOne Tools BI Publisher for JD Edwards EnterpriseOne Guide*.

---

## 4.3 Printer Information

When you submit a UBE from batch versions to a server, a data structure that contains the printer information is passed to the batch engine. Printer information is stored as a binary large object (BLOB) in the Job Control Status Master (F986110) table.

When batch jobs are submitted using report interconnects, the printer definition for the child report is inherited from the parent report. The values that are inherited are:

- Printer name

Unless the child report specification has a printer name or uses the system function to override the printer name.

- Print immediate
- Save Printer Definition Language file
- Paper type
 

Unless the child report specification has a custom override.
- Printer settings
- Number of copies
- Paper source
- OSA

Export to CSV is not an inherited functionality. 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.

## 4.4 Customizing the Location of Report Output

When you run a batch version on a server, the resulting report is saved to the PrintQueue directory as a PDF.

The default PrintQueue directory is located under the install directory on the server. If you want to customize the PrintQueue directory location, you must define the PrintQueue location in the initialization file using a valid directory name.

This example illustrates how the JDE.INI should be modified:

```
[NETWORK QUEUE SETTINGS]
OutputDirectory = /system
```

Where system is the location of the PrintQueue directory; for example, c:\system\.

---

**Note:** Any platform can use this setting to customize the PrintQueue directory location.

---

## 4.5 Printer Selections at Runtime

When you submit batch versions to a printer, the system looks first for the printer defined in RDA. If no printer is defined in RDA, the system uses the default printer defined in the JD Edwards EnterpriseOne Printer Application (P98616.) The system determines a printer based on this hierarchical structure:

1. System function.  
Do\_Initialize\_Printer called in the Report Event Rule.
2. Printer defined in RDA.  
The printer definition becomes part of the report specifications.
3. Printer defined in the JD Edwards EnterpriseOne Printer Application (P98616) using this hierarchy:
  - a. Specific report defined for the user.
  - b. Specific report defined for a user role.
  - c. Specific report defined for \*PUBLIC.

- d. All reports defined for the user.
- e. All reports defined for a user role.
- f. All reports defined for \*PUBLIC.

4. User override of defined printer at submission time.

See [Customizing the Location of Report Output](#).

Depending on the printer that is selected, the system selects the corresponding Printer Definition Language on the Advanced 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, once the job has completed processing, you can override the printer for the child report from the JD Edwards EnterpriseOne Work With Servers (P986116) application.

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 P986116 when the job has completed processing.

See "Submitting at the Command Line" in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

## 4.6 Paper Type Selections at Runtime

The paper type that you define for the selected printer is used when the job is submitted unless the report has a custom paper type defined in RDA. If a custom paper type is defined in RDA, the custom paper type is used when the job is submitted. 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 type in RDA, you cannot change the paper type at runtime.

**See Also:**

- [Working with Report Printing Administration](#).

## 4.7 Print Orientation Selections at Runtime

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

When you define a custom paper size in RDA, the orientation option 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 define for the line printer determine the orientation.

## 4.8 Export to CSV at Runtime

When you select the Export to CSV option in RDA, the definition is stored in the report specifications. This option is displayed in the Report Output Destination form 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. Both a CSV file and a PDF file are created in the PrintQueue directory when the Export to CSV option is selected.

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 RDA to enable this option for both the child and parent reports.

## 4.9 Print Settings in the Initialization Files

The Print Immediate option and the SavePDL setting are defined in the initialization files. A Printer Definition Language (PDL) file is a file that was created from a reports PDF output after being converted to a printer-specific format, such as postscript, PCL, or line printer text output.

### 4.9.1 The Print Immediate Setting

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

#### 4.9.1.1 Print Immediate--Microsoft Windows Client

Modify the following setting in the jde.ini to specify the Print Immediate functionality for batch jobs:

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

The value of TRUE sends the report output to the printer immediately after processing. FALSE holds the report output in a queue until you select Print from the Row menu on the Submitted Job Search form on the Microsoft Windows client.

When the Print Immediate setting in the initialization file is set to TRUE, all batch versions print immediately upon processing. When you want specific batch versions to print immediately, use the Print Immediate option on the Version Details form for the individual batch version. The Print Immediate option can be overridden at runtime.

See "Modifying Properties of Batch Versions" in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

#### 4.9.1.2 Print Immediate--HTML Client

Modify the following setting in the jas.ini to specify the Print Immediate functionality for batch jobs:

```
[OWWEB]
PrintImmediate=TRUE
```

When the Print Immediate setting in the initialization file is set to TRUE, all users submit all reports with Print Immediate selected. The value of TRUE sends the report output to the printer immediately after processing.

When the Print Immediate setting is set to FALSE, only reports submitted by users in the Print Immediate override section of the jas.ini file are submitted with Print

Immediate. Reports submitted by users not included in the Print Immediate override section are held in a queue until you select Print from the Row menu on the Submit Job - Submitted Job Search form on the web client. You can add Print Immediate Override to specific EnterpriseOne users and reports by adding a new section in the jas.ini file as follows:

```
[PRINTIMMEDIATE]
userid1=Report1,Report2,Report3,Report4
userid2=Report1,Report2,Report3,Report4
userid3=Report1,Report2,Report3,Report4
```

#### **4.9.1.3 Print Immediate--Interconnected Reports**

A 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.

#### **4.9.1.4 Print Immediate--Microsoft Windows Server**

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 setting in the jde.ini is set to TRUE.

---

**Note:** The Print Immediate argument for RUNUBE is supported on the Microsoft Windows Server only; this is not supported on other platforms.

---

### **4.9.2 The SavePDL Setting**

You can modify the following setting in the jde.ini to specify the SavePDL functionality on the enterprise server:

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

When you modify the SavePDL setting in the jde.ini to TRUE, both a PDF file and a Printer Definition Language (PDL) file are created in the PrintQueue directory. By default, at runtime, the Printer Definition Language File option on the Document Setup tab of the Printer Selection form is selected. However, you can override this option at runtime.

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

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

---

# Working with Report Printing Administration

This chapter contains the following topics:

- [Section 5.1, "Understanding Report Printing Administration"](#)
- [Section 5.2, "Working with the JD Edwards EnterpriseOne Printer Application"](#)
- [Section 5.3, "Setting Up Barcode Fonts"](#)
- [Section 5.4, "Understanding Multiple Code Sets for PCL"](#)
- [Section 5.5, "Designing Reports to Print on Line Printers"](#)
- [Section 5.6, "Printing Reports"](#)

## 5.1 Understanding Report Printing Administration

The JD Edwards EnterpriseOne Printer 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 JD Edwards EnterpriseOne tables that are maintained by P98616.

JD Edwards EnterpriseOne includes a number of predefined reports and batch versions, which you can use to meet the 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 JD Edwards EnterpriseOne Report Design Aid (RDA). The JD Edwards EnterpriseOne batch engine generates these reports in PDF. You can view the PDF files using Adobe Reader.

Reports must have at least one batch version before you can process the report. Batch versions do not require user interaction.

When you submit batch versions for processing, you can make some selections at runtime. These selections include:

- Data selection.
- Data sequencing.
- Location where the report processes.
- Logging capabilities to monitor processing.
- The printer to which the report is sent.

**See Also:**

- "Understanding Report Processing" in the *JD Edwards EnterpriseOne Tools Report Design Aid Guide*.
- "Submitting Batch Versions in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

## 5.2 Working with the JD Edwards EnterpriseOne Printer Application

This section provides overviews of the JD Edwards EnterpriseOne Printer Application (P98616) and null pass-through print filters and discusses how to:

- Add printers.
- Define default printers.
- Modify printers.
- Copy printers.
- Delete printers.
- Delete paper types.
- Select a print style option.
- Add null pass-through print filters.
- Search for incorrect printer records.

### 5.2.1 Understanding the JD Edwards EnterpriseOne Printer Application

JD Edwards EnterpriseOne provides a single application for defining system printers. The JD Edwards EnterpriseOne Printer Application (P98616) uses a director interface with instructions included to guide you through the setup process. From this director, you can:

- Add new printers
- Modify existing printers
- Define default printers

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

#### 5.2.1.1 Platform Information

When defining the server name and the shared name for printers in P98616, consider these platform specific guidelines:

- IBM i platform:  
library name/outqueue name

For the IBM i, 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: OutputQueue Name:DEVPRN1

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

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.

---

- Microsoft Windows platform:

\\print server name\print shared name

Examples:

Print Server Name:corptrs1

Print Shared Name:devprn1

For Microsoft Windows, 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 platform:

unix printer queue name (no slashes)

devprn16

This information must be entered in lower case.

### 5.2.1.2 Printer Definition Language

When defining the Printer Definition Language (PDL) for printers in P98616, you might be presented with additional options based on the PDL and platform type that 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 IBM i platform, options appear within a box labeled iSeries Only. Use these fields to define the IBM i encoding that the printer supports:
  - ASCII Encoding
  - EBCDIC Encoding
- When you define a PostScript or PCL printer in combination with the IBM i 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 P98616. 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 menu 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 [Understanding Pass-Through Print Filters](#).

### 5.2.1.3 Paper Types

When defining printers, you can select from predefined paper types or add new paper types from the Form menu. This table describes the paper type definitions available for new paper types:

Paper Type Definitions	Description
Paper Type	Enables you to select the type of paper that the defined printer supports, such as legal, letter, and A4.
Paper Height	Enables you to enter a value that indicates the height of the paper for the selected paper type.
Paper Width	Enables you to enter a value that indicates the width of the paper for the selected paper type.
Unit of Measure	Enables you to enter 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 the 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.

### 5.2.1.4 Default Printers

To set a printer as the default, you must select the Define Default Printer option of P98616 after adding the printer to the system. The printer can be defined as the default printer for:

- A specific version of a report.
- All versions of a specific report.
- 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.

### 5.2.2 Understanding Pass-Through Print Filters

The pass-through print filter enables you to send PDF documents directly to a print queue without converting it to a printer language format. To use a pass-through print filter, you define a custom PDL. To define a pass-through print filter, first select the Custom PDL on the Printer Setup form in P98616, 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.

This example defines the parameter string options for pass-through print filters:

```
-s script_name -l library_name -f FunctionName
```

Where -s defines the shell script name (used on UNIX to send to a printer), -l defines the shared library name (this option is the letter l, not the number 1), and -f defines the function name within that shared library to use for PDF conversion.

If the conversion program name is \*JDE PDF, no conversion occurs and the PDF file is sent to the print queue.

The parameter string for Unix is:

```
-s POSTSCRIPT_PRINTER
```

The -l and -f parameters are ignored.

### 5.2.3 Forms Used to Add Printers

Form Name	FormID	Navigation	Usage
Printers	W98616S	EnterpriseOne Life Cycle Tools, Report Management, Batch Processing Setup (GH9013), Printers. An alternative is to type P98616 in the Fast Path	Add printers, modify printers, and define default printers
Printer Setup Director	W98616AC	Click Next 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 printer shared name.

Form Name	FormID	Navigation	Usage
Printer Setup	W98616AE	Click Next on the Platform Information form when adding printers. Select a printer and click Select on the Work With Printers form when modifying printers.	Enter the location and model of the printer, paper types, default paper type, and Printer Definition Language.
Work With Printers	W98616Y	Click Modify Printer on the Printers form.	Select a printer to modify or copy.
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.

## 5.2.4 Adding Printers

Access the Printers form.

### 5.2.4.1 Platform Information

Access the Platform Information form.

**Figure 5-1 Platform Information Form**

**Printers - Platform Information**

  Previous  Next  Form  Tools

Enter a platform type for the printer, then press the Tab key.  
Enter the appropriate printer information for the platform.

**Platform Type \***  If a user selects LOCAL as the platform type, the platform type will default to NTSVR. NTSVR will encompass all NT platforms.

**Printer Information**

Print Server Name	<input type="text" value="corprts1"/>
Print Shared Name	<input type="text" value="devprn2"/>

**Example**

If a user enters:  
Server Name: corprts1  
Shared Name: devprn1  
The NT Printer name will be: \\corprts1\devprn1


**Platform Type**

Enter the type of hardware on which the database resides.

**Print Server Name**

Enter the name of the machine that receives documents from clients

**Print Shared Name**

Enter the name of the printer to which the report will be sent.

**5.2.4.2 Printer Setup**

Access the General tab on the Printer Setup form.

**Figure 5–2 Printer Setup Form - General Tab**

Printers - Printer Setup

General Details

Enter the location and the model of the printer.

Printer Name \*: \\corprts1\devprn2

Platform Type \*: NTSVR

Printer Model: LASER PRINTER

Printer Location: Training

Type a 1 in the Selected Paper Type column for each paper type you want to use with this printer. Type 1 in the Default Type column for the paper type you want to use as the default. To Add new paper types, choose New Paper Type from the Form menu.

Records 1 - 4							Customize Grid
	Select Paper Type	Default Type	Paper Type	Printer Paper Width	Printer Paper Height	UM	
○			A4	210.00	297.00	MM	
○			LEGAL	8.50	14.00	IN	
○			LETTER	8.50	11.00	IN	
○							

**Printer Name**

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

**Printer Model (Release 9.1 Update 3)**

Enter the printer model as a free-form name (for example, HP Laserjet 4700).

If you enter a PostScript, PDF, or PCL printer model, add PJL (Printer Job Language) anywhere in the field to instruct the printer to use PJL to enter the correct interpreter mode, such as PostScript, PDF, or PCL. This feature is available for printers that may require it, but is optional.

If you enter a model, you can use the Source Tray selection feature. To do so, you must add the pipe character (|) to the end of the printer model name, followed by the PostScript Printer Definition (PPD) file name. The PPD file is supplied by your printer vendor and is delivered with its drivers. You must place the file in the system/resource/PPD directory on any Enterprise Server or Windows client from which you want to enable this feature.

For example, you want to use the Source Tray selection feature with a Xerox WorkCentre 5638 printer. The PPD file is named xrwc5638.ppd. You place this file in the system/resource/PPD/xrwc5638.ppd location on your Enterprise server or Windows client.

For this printer model, you will use PJL to instruct the printer to enter PostScript mode because it will not properly interpret the PostScript code in some cases. If the printer

does not enter PostScript mode, then the printer prints pages of PostScript programming source code instead of printing your actual document.

Therefore, for this example, you enter the following information in the Printer Model field:

PJL | XRWG5638.PPD

---

**Note:** All alphabetic characters in the PPD filename must be either all uppercase or all lowercase on the file system; EnterpriseOne defaults to all uppercase characters in the data field. However, for Unix systems, EnterpriseOne will search for and find all lower case file names. It does not matter if there are spaces around the pipe character.

---

#### **Printer Location**

Enter the physical location of the printer. This entry is a free-form field for CNC use.

#### **Attachments**

Double-click this field to select the paper type to be used with the defined printer. A check mark appears next to selected paper types.

#### **Default Type**

Select a user defined code (UDC) (98 | FP) that indicates the default paper type. Values are:

- 1:** Indicates that the paper type is the default.
- 0:** Indicates that the paper type is available for use with the defined printer but is not the default

#### **Paper Type**

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

#### **Printer Paper Width**

Enter a value to indicate 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**

Enter a value to indicate 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**

Select a UDC (00 | UM) that indicates the unit of measure that is used to define the width and height of the defined paper type.

#### **5.2.4.3 Printer Setup**

Access the Details tab on the Printer Setup form.

**Figure 5–3 Printer Setup Form - Details Tab**

Printers - Printer Setup

General Details

Choose the printer definition languages(PDL) that the printer supports. If you choose the Line Printer option, also enter paper dimensions and line parameters in the Line Printers box.

For AS400s, choose the type of encoding to use for the printer. Click the EBCDIC Encoding option if the printer is an EBCDIC line printer.

**Printer Definition Language**

**Default**

<input type="checkbox"/> PostScript	<input checked="" type="radio"/>
<input checked="" type="checkbox"/> PCL	<input checked="" type="radio"/>
<input type="checkbox"/> Line Printer	<input type="radio"/>
<input type="checkbox"/> Custom	<input type="radio"/>

Type a 1 in the Selected Paper Type column for each paper type you want to use with this printer. Type 1 in the Default Type column for the paper type you want to use as the default. To Add new paper types, choose New Paper Type from the Form menu.

Records 1 - 4							Customize Grid
	Select Paper Type	Default Type	Paper Type	Printer Paper Width	Printer Paper Height	UM	
<input type="radio"/>			A4	210.00	297.00	MM	
<input type="radio"/>			LEGAL	8.50	14.00	IN	
<input checked="" type="radio"/>		<input checked="" type="radio"/>	LETTER	8.50	11.00	IN	
<input type="radio"/>							

**Printer Definition Language**

Select the name of the 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 P98616. Only users with knowledge of building parameter strings for printers should use this option.

---

**Maximum Number of Paper Sources**

Enter the maximum number of paper trays available on the defined printer. This field is available only when PostScript or Custom PDL is selected.

**Default Paper Source**

Enter the output tray to be used for a specific batch job. This field is available only when PostScript or Custom PDL is selected.

**Characters per Inch**

Enter the number of characters per horizontal inch supported by the defined printer. This field is 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**

Enter the number of columns per page supported by the defined printer. This field is 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**

Enter the number of lines per inch supported by the defined printer. This field is 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**

Enter the number of lines per page supported by the defined printer. This field is only available when the Line Printer PDL is selected. For an 8 1/2 x 11 inch page the lines per page value is 66.

**Printer Paper Width**

Displays the paper width based on the value that you enter in the Columns per Page field. This field is populated by the system and only appears when the Line Printer PDL is selected.

**Printer Paper Height**

Displays the paper height based on the value that you enter in the Line per Inch field. This field is populated by the system and only appears when the Line Printer PDL is selected.

## 5.2.5 Defining Default Printers

Access the Default Printer Revisions form.

**Figure 5–4 Default Printer Revisions Form**

**Printers - Default Printer Revisions**

   Tools

Users may add several default printers for a valid user, host name, and environment combination. Only one record can be active at one time. If users choose "LOCAL" as the host name, it will be converted to "WinClient".

User/Role *	*PUBLIC
Report Name	*ALL
Version Name	*ALL
Environment	*ALL
Printer Name *	\corprts1\devprn1
Host Name	*ALL
Object Status	AV Active


**User/Role**

Enter the user ID or role that had permissions to use the printer. \*PUBLIC gives permissions to all users.

**Report Name**

Enter 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**

Enter 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**

Enter 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**

Enter the name of the printer defined in P98616 to be used as the default.

#### **Host Name**

Enter 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**

Enter a UDC (H98|ST) that indicates whether the default printer is active or inactive.

### **5.2.6 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.

### **5.2.7 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.

### **5.2.8 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.

### **5.2.9 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.

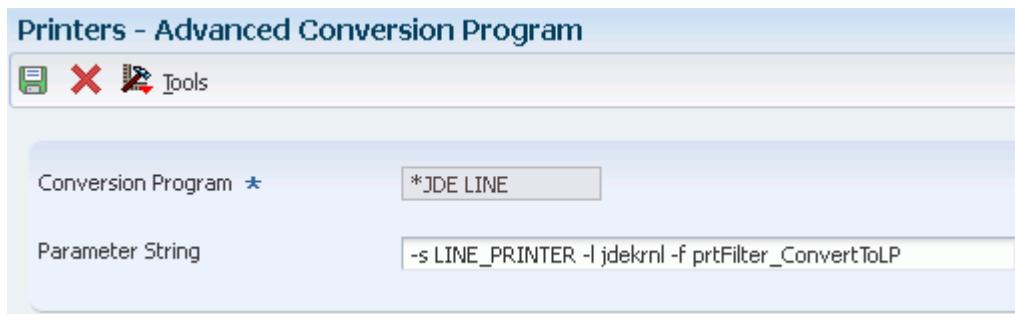
### 5.2.10 Selecting a Print Style Option (Release 9.1 Update 3)

- Simplex - select this option if you want to print as a one-sided document.
- Duplex (Duplex-No tumble) - select this option if you want to print double-sided, along the long edge of the paper. This is for the Portrait layout
- Tumble (Duplex -Tumble) - select this option if you want to print double-sided along the short edge of the paper. This is for Landscape layout.

### 5.2.11 Adding Pass-through Print Filters

Access the Advanced Conversion Program form.

**Figure 5–5 Advanced Conversion Program form**



#### Conversion Program

Select the name of a conversion program (null pass-through print filter) using the visual assist.

#### Parameter String

Displays 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.

### 5.2.12 Searching for Incorrect Printer Records

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

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 [Modifying Printers](#).

## 5.3 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.

### 5.3.1 Understanding Barcode Fonts

JD Edwards 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.

---

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

Form Name	FormID	Navigation	Usage
Work With Bar Code Font	W986166A	Type P986166 in the Fast Path.	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.

### 5.3.3 Setting Up Printers to Use Barcode Fonts

Access the Bar Code Support Revisions form.

**Figure 5–6 Bar Code Font Revisions form****Printer Name**

Use the visual assist to select a printer to be used for printing barcode fonts.

**Printer Definition Language**

Select the printer definition language used by the printer selected. Options include PostScript and PCL.

**True Type Font Name**

Click the True Type Font button to select a font, such as the BC C39 3 to 1 Medium barcode font. The bar code true type font must reside in the Microsoft Windows font directory as well as the <system/resource/truetype> directory on the JD Edwards EnterpriseOne server where the report will run.

**Printer Font Name**

Enter the name of the printer font to be used.

**Symbol Set ID**

Enter 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.

**5.3.3.1 PCL Printer**

If you use a PCL printer, you must purchase the bar code soft font for the printer from a third-party vendor. Order Code39 Fonts, UNIX (PFA) or UNIX (PFB) to get PostScript soft fonts (this applies even if your are not running a UNIX enterprise server.) Download the soft fonts to your printer using a font download program. After you download the soft fonts to your printer, use the Bar Code Font Support (P986166) application to set up the barcode in JD Edwards EnterpriseOne.

**5.3.3.2 PostScript Printer**

If you use a PostScript printer, the Barcode True Type Font file must be installed in the appropriate location, as indicated here:

- Microsoft Windows Server or Client

Install the bar code true type font in both C:\WINNT\Fonts and in your EnterpriseOne system directory path: system\Resource\truetype.

- UNIX Server

Install the bar code true type font under the path: \$SYSTEM/resource/truetype.

- IBM i Server

Install the bar code true type font on the IFS in your system directory under the path: SYSTEM/resource/truetype.

#### **5.3.3.3 Printer Configuration Troubleshooting**

- If your printer is connected to a Microsoft Windows print server with JD Edwards EnterpriseOne running on a UNIX server, the queue must be configured as a UNIX remote queue that points to the Microsoft Windows server and the Microsoft Windows printer server must be running the LPD printer service daemon.
- When a UNIX remote queue is a UNIX queue on an AIX box, it must be configured as a raw queue (that is, it cannot have an AIX print processor on it.)
- For the setting in P986616 to take effect, a True Type Font (TTF) font must map to the PCL font and the TTF font must be used for the characters in the report specifications that are intended for the barcode; otherwise, the substitution of TTF to PCL soft font will not take place.

#### **5.3.4 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.

#### **5.3.5 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.

#### **5.3.6 Deleting Barcode Support Information from Printers**

Access the Work With Bar Code Font form.

1. Select the printer for which you want to delete barcode information and click Delete.
2. On the Confirm Delete form, click OK.

### **5.4 Understanding Multiple Code Sets for PCL**

This section discusses:

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

## 5.4.1 Multiple Code Sets for PCL Printing

The JD Edwards 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 the 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.

This table describes scenarios that you should consider for defining the PRTPCLSymbolSet:

Scenario	Description
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.

## 5.4.2 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.

### 5.4.2.1 Code Page Order of Precedence

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

- Checks the Bar Code font support (F986166) table to determine if the report contains barcode information.  
If the report includes barcode information, it prints the report using a barcode 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.

#### 5.4.2.2 Symbol Set Order of Precedence

After the code set is determined, the PCL print filter uses rules to determine the symbol set. This table indicates the rules for the PCL print filter:

Print Filter	Rule
Checks to determine if a symbol set is specified in the Bar Code font support 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.

## 5.5 Designing Reports to Print on Line Printers

This section provides overviews of reports designed to print on line printers and remote IBM i line printers used to print multiple copies of reports, lists the prerequisite, and discusses how to:

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

### 5.5.1 Understanding Reports Designed to Print on Line Printers

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

- 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.

These font sizes are available on most line printers; most line printers support only these sizes:

Font	Characters per Inch (CPI)	Suggested Font Point Size in RDA
Courier New	10	12.00
Courier New	12	10.00
Courier New	17	7.06
Courier New	20	6.00

- Modify the field width.

Because font properties affect the size of the report fields, you might need to adjust field 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 "Modifying the Appearance of Report Objects" in the *JD Edwards EnterpriseOne Tools Report Design Aid Guide*.

## 5.5.2 Modifying Reports to Print on Line Printers

You can print reports to a line printer. Before modifying reports to print on line printers, ensure that you perform one of these steps:

- Create a batch version of a 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 on line printers.

Select a batch version that is checked out and launch JD Edwards EnterpriseOne 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. Align fields as needed using the Align option on the Layout menu.
9. Save the version.

### 5.5.2.1 Object Alignment

Sometimes it is difficult to get objects to align to each other properly. Use these steps to align objects:

1. Select all of the objects that you want to align while holding the control key down. The last field that you choose is the field that will be used to align the other fields.
2. From the layout menu, choose Align.
3. To box, select the current section option, and then in the Top-to-Bottom box, select the Bottom Edges option.
4. Save your report.

### 5.5.2.2 Group Sections

By default, the batch engine resizes the heights of your objects to be the correct height for the number of lines per inch for your printer. This works very well for columnar

and tabular sections. Due to the way group sections work, it can be difficult to make vertical positioning work because all objects in a group section have various dependency rules that determine spacing. (Actually, in all sections, but this manifests as a problem for line printing only in group sections.)

### 5.5.3 Defining the Printer for Line Printing

The CPI, LPI, CPP, and LPP settings of your printer are used to set a PDF page size for your job when sent to that printer.

The page size setting in RDA is merely a visual guide and has *no effect* on your report's page size unless you check the *Custom* check box. When the *Custom* option is selected, the report will *always* get the custom page size from the report specifications no matter what the printer has defined for a page size. This table shows some common page sizes and how they map to CPI, CPP, LPI, and LPP:

Page Width (Inches)	Page Height (Inches)	CPI	LPI	CPP	LPP
11	8.5	10	6	110	51
8.5	11	10	6	85	66
14	8.5	10	6	140	51
8.5	14	10	6	85	84

#### 5.5.3.1 Line Printing

When you send a job to a line printer, consider the following:

- All dot matrix printers are line printers.
- To define custom paper size, you need to know CPI, CPP, LPI, LPP.

The formula for defining paper size is:

CPI \* custom inches wide = CPP, LPI \* custom inches = LPP tall.

- Calculation of CPP and LPP depends on the paper size that you are using for your line printer.

Some common paper sizes that are available for line printers are 8.5 x11 inches; 13.2 x 11 inches, 13.2 x 7.5 inches, and 13.2 x 8.5 inches.

- For example, CPP and LPP calculations for paper size 13.2 x 7.5 inches at 10 CPI and 6 LPI would be 13.2 inches \* 10 CPI = 132 CPP and 7.5 inches \* 6 LPI = 45 LPP. (Think of this as 13.2 inches per page width \* 10 characters per inch = 132 characters per page width.)
- Similarly, for 11 inches wide by 8.5 inches tall, at 12 CPI and 8 LPI, the calculations would be: 11 inches wide \* 12 CPI = 132 CPP; 8.5 inches tall \* 8 LPI = 68 LPP.

- Line printers cannot show colors, bold, italic, or underlines or boxes for column header identification.

#### 5.5.3.2 Epson and IBM Line Printer Settings

Generally all line printers support both 10 CPI and 12 CPI, but it is dependent on how you set up the printer. If the printer is going to be used to print only Western European (for example, CP 1252), and it supports either IBM PPDS line printer escape

codes or Epson ESC/P line printer escape codes, you can define a conversion string that uses a different entry point that uses those escape codes. You actually send the escape code to set the line printer to the correct CPI and LPI settings. Use one of these examples to print the characters in CP1252 between 0xA1 and 0xFF to a line printer:

IBM PPDS conversion string:

```
-s LINE_PRINTER -1 jdekrnl -f prtFilter_ConvertToPPDS
```

Epson ESC/P conversion string:

```
-s LINE_PRINTER -1 jdekrnl -f prtFilter_ConvertToESCP
```

## 5.6 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.

### 5.6.1 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.

Within JD Edwards EnterpriseOne, you have the option of viewing the PDF output using Adobe Reader or sending the report directly to a printer. You can also print the report from Adobe Reader. When you send the report to a printer, the system uses a conversion filter to transform the PDF file into a PDL format:

- PCL 5
- PostScript level 3
- 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 RDA.  
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 Printer Definition (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.

### 5.6.2 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, *E812\PrintQueue*, of the client install on the workstation.

When you view the report from the web client, the system copies the PDF file from the JD Edwards EnterpriseOne server to a temp directory on the HTML (JAS) server. The temp directory is defined in the *jas.ini* file. The PDF is then sent from the HTML server to the user's web browser. PDF files are deleted when you sign off of the web client.

In the *jas.ini* file, modify the *JDENET* setting, as illustrated in this example:

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

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.

See [Customizing the Location of Report Output](#).

### 5.6.3 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 JD Edwards EnterpriseOne Report Design Aid (RDA). 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 Reader. The PDF file is stored in the *PrintQueue* subdirectory of the client install on the workstation.

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 *E812\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 RDA, 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.
- Sends the PDL file to the defined printer.

See "Submitting Batch Versions from the Microsoft Windows Client" in the *JD Edwards EnterpriseOne Tools Batch Versions Guide*.

#### 5.6.4 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 in the PDF file.

#### 5.6.5 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	<p>Specifies whether the system automatically prints the report after processing is complete. Values are:</p> <p>TRUE.</p> <p>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.</p> <p>FALSE.</p> <p>The system processes the report on the server, but does not automatically print the report. Users must access the Submitted Job Search form to manually print the report.</p>
SaveOutput	<p>Specifies whether the system saves or deletes the output after the user views or prints the job. Values are:</p> <p>TRUE.</p> <p>The system saves the output after it has been viewed or printed.</p> <p>FALSE.</p> <p>The system deletes the output after it has been viewed or printed.</p>

The jas.ini settings control whether reports print immediately:

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

This table describes the jas.ini setting:

Setting	Description
PrintImmediate	<p>Specifies whether the system automatically prints the report after processing is complete. Values are:</p> <p>TRUE.</p> <p>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.</p> <p>FALSE.</p> <p>The system processes the report on the server, but does not automatically print the report. Users must access View Job Status–Work With Servers form to manually print the report.</p>

---

## Working with Output Stream Access

This chapter contains the following topics:

- [Section 6.1, "Understanding OSA"](#)
- [Section 6.2, "Creating OSA Libraries"](#)
- [Section 6.3, "Creating and Associating OSA Interfaces"](#)

### 6.1 Understanding OSA

You can use OSA interfaces to output to third-party software programs. OSA interfaces enable the third-party program to process and format JD Edwards EnterpriseOne data concurrently with the processing of a UBE. OSA interfaces must be predefined, typically by a representative of the third-party product that you are using. Several interfaces may exist for one program, depending on the section types included in the report and the desired output.

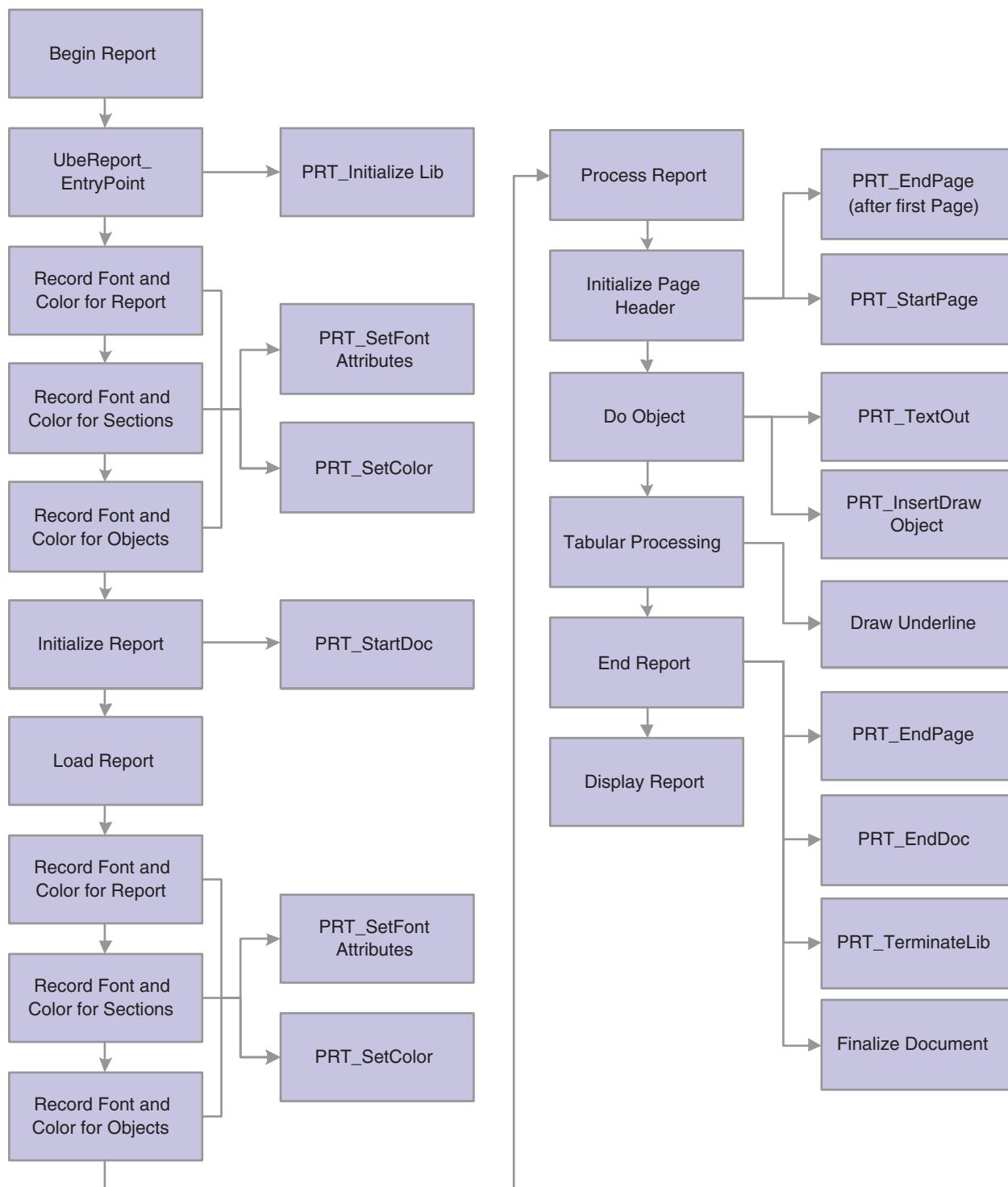
OSA uses its own set of commands from a third-party library.

Benefits of using OSA are:

- Employs the formatting options of the target software program.
- Employs the processing power of the target software program.

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:

**Figure 6–1 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.

## 6.2 Creating OSA Libraries

This section discusses:

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

### 6.2.1 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 the business requirements.

JD Edwards 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.

#### 6.2.1.1 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);
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);
```

### 6.2.2 Function Parameters

Function parameters define:

- Start document parameters
- Set font parameters

- Set color parameters
- Start page parameters
- Text out parameters
- Insert draw object parameters
- Draw underline parameters
- End page parameters
- End document parameters
- Finalize document parameters

#### **6.2.2.1 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
```

#### **6.2.2.2 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
```

#### **6.2.2.3 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
```

#### **6.2.2.4 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
```

#### **6.2.2.5 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
```

#### **6.2.2.6 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
```

#### 6.2.2.7 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
```

#### 6.2.2.8 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
```

#### 6.2.2.9 Defining End Document Parameters

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

```
OSA_REPORT_INFO *pOSAReportInfo
```

#### 6.2.2.10 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
```

### 6.2.3 OSA Documents

Within the OSAResponseInfo 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 OSAResponseInfo member for szOSAFileName is copied to the corresponding UBEVar member. When a job has finished processing, the UBEVar structure is updated into the Job Control Status Master (F986110) table for that job.

### 6.2.4 Include Files

The structure and function definitions required for functions interfacing through OSA are available in the JDEOSA.H file, which is located in the system\include directory.

### 6.2.5 File Locations and Names

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

1. If the library name, as defined in the OSA Interface Definition (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
Microsoft Windows	libname + .dll
Solaris, Linux, AIX, HP-UX	lib + libname + .so
IBM i	libname

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

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. The OSASample output has the extension .osa.

## 6.2.6 OSASample Source Code

OSASample includes three components:

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

### 6.2.6.1 OSAStruct.h

This example shows OSAStruct.h source code:

```
#ifndef OSASAMPLE_DEF_HPP
#define SASAMPLE_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
```

### 6.2.6.2 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
#define CDECL
#define APIEXPORT
#endif
#define CLASSEXPORT APIEXPORT
#define 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);
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

```

### 6.2.6.3 OSASAMPLE.c

```
#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
pOSAReportInfo)
{
/* 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);
    jde 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->szEnv);
    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***** 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***** ITEM INFO *****\n\n);
    fprintf(pOSAStruct->fpOutput, \t\tOccurrence Count: %d\n, pOSAItemInfo->
ulOccurrenceCount);
    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\tDisplayStyle: %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,
POSALINK_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 (OSA_REPORT_INFO pOSAReportInfo, POSASAMPLE_STRUCT
pOSAStruct)
{
/* Formulate the output file name based on information from Report Info. */
strcpy(pOSAReportInfo->szOSAFileName, pOSAReport Info->szUBEFileName);
#if defined JDENV_AS400
/* On IBM i, 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 IBM i, just replace the PDF file extension with OSA. */
if( !strstr( pOSAReportInfo->sz OSAFileName,.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(pOSAResponse->szLogMessage, Could not
        open OSA file: %s\n,
        pOSAResponse->szOSAFileName);
return;
}
return;
/*-----*/
/* Name: OSASample_StartDoc */
/* Parameters: OSA_REPORT_INFO* */
/* Exceptions: None */
/* ReturnValue: None */
/* Description: Open the output file, */
/* Output Report, Section and Object */
/* properties. */
/*-----*/
EXTERNC APIEXPORT void CDECL OSASample_StartDoc(OSA_REPORT_INFO*pOSAResponse)
{
    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(!pOSAResponse )
    {return;
    }
    /* Allocate memory to hold severity value.
    Deallocated in OSASample_EndDoc */
    if (!pOSAResponse->pnLogMessageSeverity)
    {
        pOSAResponse->pnLogMessageSeverity = malloc(sizeof( unsigned short));
    }
    if (pOSAResponse->pnLogMessageSeverity)
    {
        pOSAResponse->pnLogMessageSeverity[0] = 0;
    }
    /* Create the common structure for passing values between functions,
    if it has not been created before this point. */
    if (!pOSAResponse->pExternalDataPointer)
    {
        pOSAstruct = malloc(sizeof( OSASAMPLE_STRUCT));
        if (pOSAstruct)
        {
            memset(pOSAstruct, 0, sizeof(OSASAMPLE_STRUCT));
        }
        else
        {
            strcpy(pOSAResponse->szLogMessage, OSA: Could not allocate External Data
Pointer.\n);
            /* Set the correct severity to error message severity */
            if (pOSAResponse->pnLogMessageSeverity)
            {
                *(pOSAResponse->pnLogMessageSeverity) = (unsigned short)1;
            }
        }
        /* Record the pointer as the external data pointer in Report Info. */
        pOSAResponse->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*/

```

```

/* 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
{
    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(OSA_REPORT_INFO
pOSAReportInfo,
OSA_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(pOSAResponse->szLogMessage, OSA: Could not allocate External
Data Pointer.\n);
/* Set the correct severity to error message severity */
if (pOSAResponse->pnLogMessageSeverity)
{
d*(pOSAResponse->pnLogMessageSeverity) = (unsigned short)1;
}
}
/* Record the pointer as the⇒
external data pointer in Report Info.*/
pOSAResponse->pExternalDataPointer=p⇒
OSAStruct;
}
/* Output Report Info */
pOSAStruct=pOSAResponse->pExternalDataPointer;
/* Create output file if it has not been created yet */
if (!pOSAStruct->fpOutput)
{
OSA_OpenOutputFile (pOSAResponse, pOSAStruct);
if (pOSAResponse->pnLogMessageSeverity[0] > 0)
return;
}
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAResponse->szLogMessage, OSA: No Output File pointer at Set Font.\n);
/* Set the correct severity to error message severity */
if (pOSAResponse->pnLogMessageSeverity)
{
*(pOSAResponse->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(OSA_REPORT_INFO
pOSAResponse,
unsigned long int zColorRef)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely), then return */
if(!pOSAResponse )
{
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(!pOSAResponse->pExternalDataPointer)

```

```

{
strcpy(pOSAResponse->szLogMessage, OSA: No External Data Pointer at Set
Color.\n);
/* Set the correct severity to error message severity */
if (pOSAResponse->pnLogMessageSeverity)
{
*(pOSAResponse->pnLogMessageSeverity) = (unsigned short)1;
}
return;
}
/* Output Report Info */
pOSAStruct=pOSAResponse->pExternalDataPointer;
/* Check for valid file pointer */
if (!pOSAStruct->fpOutput)
{
strcpy(pOSAResponse->szLogMessage, OSA: No Output File pointer at Set
Color.\n);
/* Set the correct severity to error message severity */
if (pOSAResponse->pnLogMessageSeverity)
{
*(pOSAResponse->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(OSA_REPORT_⇒
INFO
pOSAResponse,
OSA_OBJECT_INFO pOSAOBJECTInfo)
{
POSASAMPLE_STRUCT pOSAStruct = NULL;
/* If OSA does not provide the needed parameter (Highly unlikely),
then return */
if(!pOSAResponse )
{
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(!pOSAResponse->pExternalDataPointer)
{
strcpy(pOSAResponse->szLogMessage, OSA: No External Data Pointer at Text
Out.\n);
/* Set the correct severity to error message severity */
if (pOSAResponse->pnLogMessageSeverity)
{
*(pOSAResponse->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(OSA_REPORT_INFO
pOSAReportInfo,
OSA_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 (!p=>
OSAStruct->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(OSA_REPORT_INFO
pOSAReportInfo,
OSA_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;
}

```

## 6.3 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.

### 6.3.1 Understanding OSA Interfaces

Before reports can be output using OSA, you must define the interface. Typically, once defined, OSA interfaces 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. OSA interfaces can be associated in the same way as default printers with:

- Environments
- Hosts
- Users or roles

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

#### 6.3.1.1 Using the System Hierarchy to Resolve Priority Conflicts

The general hierarchy that the system uses to resolve OSA interfaces that are associated with more than one report or version is illustrated in the following table. The system uses the same hierarchy for each user or role, processing in this order:

1. Username
2. Role
3. \*PUBLIC

Report	Version	Environment	Host
report	version	environment	hosttype
report	version	environment	*ALL
report	version	*ALL	hosttype
report	version	*ALL	*ALL
report	*ALL	environment	hosttype
report	*ALL	environment	*ALL
report	*ALL	*ALL	hosttype
report	*ALL	*ALL	*ALL
*ALL	*ALL	environment	hosttype
*ALL	*ALL	environment	*ALL
*ALL	*ALL	*ALL	hosttype
*ALL	*ALL	*ALL	*ALL

### 6.3.2 Forms Used to Create and Associate OSA Interfaces

Form Name	FormID	Navigation	Usage
Output Stream Access Setup	W986168F	Type P986168 in the Fast Path.	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.

### 6.3.3 Creating OSA Interface Definitions

Access the Output Stream Access Interface Definition Revisions form.

**Figure 6–2 Output Stream Access Interface Definition Revisions Form**

**Output Stream Access Interface Definition Revisions**

   Tools

Use this form to define library and function names for the ten given execution points. For each execution point, you can enter both the library name and function name, or leave both blank.

Records 1 - 10				Customize Grid	  
	Execution Point	Output Stream Access Library Name	Output Stream Access Function Name		
<input type="radio"/>	Start Document	osasample	OSASample_StartDoc		
<input type="radio"/>	Set Font	osasample	OSASample_SetFont		
<input type="radio"/>	Set Color	osasample	OSASample_SetColor		
<input type="radio"/>	Start Page	osasample	OSASample_StartPage		
<input type="radio"/>	Text Out	osasample	OSASample_TextOut		
<input type="radio"/>	Insert Draw Object	osasample	OSASample_DrawObject		
<input type="radio"/>	Draw Underline	osasample	OSASample_DrawUnderline		
<input type="radio"/>	End Page	osasample	OSASample_EndPage		
<input type="radio"/>	End Document	osasample	OSASampleEndDoc		
<input checked="" type="radio"/>	Finalize Document	osasample	OSASample_FinalDoc		

**Output Stream Access Interface Name**

Enter a unique name that identifies a set of external functions that can receive and process information during execution. JD Edwards 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**

Enter the shared libraries that contain functions that use the data provided through the OSA interface.

**Output Stream Access Function Name**

Enter the 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.

### 6.3.4 Associating an OSA Interface with an Object

Access the Output Stream Access Interface Usage Revisions form.

**Figure 6-3 Output Stream Access Interface Usage Revisions Form**

**Output Stream Access Interface Usage Revisions**

Tools

Use this form to define usage information related to interface names. The Interface Usage is optional. If you choose to complete only some of the fields, default values of \*ALL or \*PUBLIC will fill the remaining fields.

Output Stream Access Interface Name	OSASample
-------------------------------------	-----------

Records 1 - 1

	Report Name	Version	Environment Name	Host Name	User/Role	Usage Status
<input type="checkbox"/>	R01401	ZJDE0001	JDV900	LOCAL	CK5713835	AV

Customize Grid

**Output Stream Access Interface Name**

Enter the OSA interface name to associate with an object. Use the visual assist to select a valid interface name.

**Report Name**

Enter the name of the report to associate with the OSA interface. \*ALL indicates all reports.

**Version**

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

**Environment Name**

Enter the location of the report and batch version specifications.

**Host Name**

Enter the name of the server that processes the defined batch version.

**User/Role**

Enter the user ID or role with permissions to use the OSA interface. \*PUBLIC gives permissions to all users.

**Usage Status**

Select a user-defined code (UDC) (H98|ST) that indicates whether the OSA interface is active or not active.



---

---

# Glossary

**barcode**

An optical machine-readable representation of data, which shows information about the object to which it is attached.

**batch version**

In JD Edwards EnterpriseOne, the way to process a report. Batch versions can be run locally on the Microsoft Windows client or submitted from any client to run on a server.

**Comma Separated Values (CSV)**

A specially formatted plain text file that stores spreadsheet or basic database-style information in a simple format, with one record on each line, and each field within that record separated by a comma.

**Output Stream Access (OSA)**

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

**Portable Document Format (PDF)**

A worldwide standard for capturing and reviewing information from almost any application on any computer system and sharing it with virtually anyone, anywhere.

**PostScript**

A worldwide printing and imaging standard. JD Edwards supports PostScript level 3.

**printer command language (PCL)**

A page description language (PDL) developed by HP as a printer protocol that has become an industry standard. JD Edwards supports PCL 5.

**printer definition language (PDL) file**

A file that is created from a report's PDF output after being converted to a printer-specific format such as PostScript, PCL, or line-printer text output.



---

---

# Index

## A

Advanced Conversion Program form, 5-13

## B

Bar Code Support Revisions form, 5-14

barcode fonts

- copying information for new printers, 5-16
- deleting support information from printers, 5-16
- modifying printer information, 5-16
- setting up, 5-14
- understanding, 5-14

barcode support information, deleting from printers, 5-16

batch jobs

- submitting on Microsoft Windows client, 4-2
- submitting using report interconnects, 4-4

batch versions

- designing to print on line printers, 5-18
- locating PDFs when run on the Microsoft Windows client, 5-22
- locating PDFs when run on the web client, 5-22
- printing, 5-1
- running locally on the Microsoft Windows client, 5-22
- running on the server, 5-22
- submitting using report interconnects, 4-4
- understanding submission of, 5-21

Batch Versions (P98305), 4-1

## C

code page, order of precedence for PCL printing, 5-17

comma separated value filesCSV, 4-4

conversionsnull pass-throughprint filters, 5-13

CSV

- defining at runtime, 3-3, 4-4
- defining in batch versions, 3-3
- defining in Report Design Aid, 3-6
- defining in report templates, 3-3
- design recommendations, 3-3
- exporting to, 3-3
- files created when exporting to, 3-4

## D

Default Printer Revisions form, 5-11

default printers

- defining, 5-11
- understanding, 5-4

device context, using to create PDF files, 5-21

draw underline parameters, defining, 6-5

## E

end document parameters, defining, 6-5

end page parameters, defining, 6-5

## F

finalize document parameters, defining, 6-5

function parameters

- defining draw underline parameters, 6-5
- defining end document parameters, 6-5
- defining end page parameters, 6-5
- defining finalize document parameters, 6-5
- defining insert draw object parameters, 6-4
- defining set color parameters, 6-4
- defining set font parameters, 6-4
- defining start document parameters, 6-4
- defining start page parameters, 6-4
- defining text out parameters, 6-4
- understanding, 6-3

function signatures, understanding, 6-3

## H

hierarchy

printers, 3-2

## I

IBM i

adding printers, 5-2

include files, 6-5

initialization files

jas.ini, 5-23

jde.ini, 5-23

modifying print settings, 5-23

insert draw object parameters, defining, 6-4

## J

---

jas.ini  
defining the Print Immediate option, 4-5  
modifying PrintImmediate setting, 5-23  
jde.ini  
defining the Print Immediate option, 4-5  
defining the SavePDL file option, 4-6  
modifying PrintImmediate and SaveOutput  
settings, 5-23  
JDEOSA.H file, 6-5

## K

---

K2DoInitPrinter, 3-2

## L

---

line printers  
designing reports to print on line printers, 5-18  
modifying reports to print on, 5-19  
logging, activating at runtime, 5-22

## M

---

Microsoft Windows client  
locating log files, 5-22  
locating PDF files, 5-22  
running batch versions locally, 5-22  
Microsoft Windows platform, adding printers, 5-3  
multiple code sets  
understanding for PCL, 5-17

## N

---

null pass-through print filters  
adding, 5-13  
understanding, 5-5

## O

---

order of precedence, for PCL printing, 5-17  
orientation, specifying at runtime, 4-4  
OSA  
associating interfaces with objects, 6-24  
benefits of using, 6-1  
creating interface definitions, 6-23  
naming and locating files, 6-5  
resolving priority conflicts, 6-22  
retrieving documents, 6-5  
understanding, 3-5, 6-1  
understanding function parameters, 6-3  
understanding function signatures, 6-3  
understanding interfaces, 6-22  
understanding libraries, 6-3  
OSA documents, retrieving, 6-5  
OSA file names, 6-6  
OSA interfaces  
associating with objects, 6-24  
creating definitions, 6-23  
understanding, 6-22

## OSA Libraries

naming and locating files, 6-5  
understanding, 6-3

### OSASample source code

components, 6-6  
OSASample.c example, 6-7  
OSASample.h example, 6-6  
OSAStruct.h, 6-6

### OSASample.c, source code example, 6-7

### OSASample.h, source code example, 6-6

### OSAStruct.h, source code example, 6-6

### output

defining in jde.ini and jas.ini, 5-23  
defining PrintQueue directory, 4-3  
submitting batch jobs locally on the Microsoft  
Windows client, 4-2

understanding, 2-1

### output management, understanding, 2-2

### Output Stream AccessOSA, 3-5

## P

---

### P98616, Printer Application, 3-2

### paper

selecting types, 3-2  
selecting types at runtime, 4-4

### paper types

defining in Report Design Aid, 3-5  
deleting, 5-12  
selecting at runtime, 4-4  
understanding, 3-2, 5-4

### PDF

locating when batch versions are run on the  
Microsoft Windows client, 5-22

locating when batch versions are run on the web  
client, 5-22

### PDL

defining the SavePDL option, 4-6  
understanding, 5-3

### PDL

See also Printer Definition Language, 5-3

### Platform Information form, 5-6

### platform information, defining, 5-2

### print filters

adding null pass-through filters, 5-13  
understanding null pass-through filters, 5-5

### Print Immediate option

defining for all batch versions, 4-5  
defining for individual batch versions, 4-5

### print orientation, specifying at runtime, 4-4

### print properties

modifying, 3-1  
understanding, 3-1

### print settings, understanding the order of precedence for PCL printing, 5-17

### Print Setup form, 3-5

### Printer Application, P98616, 3-2

### Printer Application, understanding, 5-2

### Printer Definition Language, 5-3

### Printer Definition LanguagePDL, 4-6

printer information  
copying barcode information for new printers, 5-16  
modifying for barcode fonts, 5-16  
storing and passing, 4-2  
printer records, searching for incorrect records, 5-13  
Printer Search & Select form, 3-5  
Printer Setup form, 5-7  
printers  
adding for IBM i, 5-2  
adding for Microsoft Windows, 5-3  
adding for UNIX, 5-3  
copying, 5-12  
copying barcode information for new printers, 5-16  
customizing the location of a report output, 4-3  
defining default printers, 5-11  
defining in Report Design Aid, 3-5  
defining paper types, 5-4  
defining platform information, 5-2  
defining the Printer Definition Language (PDL), 5-3  
deleting, 5-12  
deleting barcode support information, 5-16  
deleting paper types, 5-12  
designing reports to print on line printers, 5-18  
determining based on hierarchical structure, 3-2, 4-3  
exporting to CSV at runtime, 4-4  
modifying, 5-12  
modifying barcode information, 5-16  
modifying reports to print on line printers, 5-19  
modifying settings in initialization files, 5-23  
overriding designated printers, 3-1  
overriding print-time characteristics, 5-23  
resolving, 4-3  
searching for incorrect printer records, 5-13  
specifying print orientation at runtime, 4-4  
storing and passing printer information, 4-2  
understanding defining defaults, 5-4  
understanding print properties at runtime, 4-1  
Printers (P98616), 3-1  
Printers form, 5-6  
PrintImmediate  
defining in the jas.ini, 5-23  
defining in the jde.ini, 5-23  
printing administration, understanding, 5-1  
PrintQueue  
locating PDF files, 5-22  
priority conflicts, using the system hierarchy to resolve, 6-22

## R

---

Report Design Aid  
defining batch applications to export to CSV, 3-3  
defining batch versions to export to CSV, 3-3  
report output  
customizing the location of, 4-3  
report output, understanding, 2-1

reports  
designing to print on line printers, 5-18  
printing, 5-1  
runtime  
activating logging, 5-22  
determining printer based on hierarchical structure, 4-3  
exporting to CSV, 3-3, 4-4  
overriding OSA interface, 6-22  
overriding printer, 3-2  
reading the jde.ini and jas.ini, 2-2  
selecting paper types, 4-4  
selecting print orientation, 4-4

## S

---

SaveOutput, defining in the jde.ini, 5-23  
SavePDL file option, defining, 4-6  
servers, running batch versions, 5-22  
set color parameters, defining, 6-4  
set font parameters, defining, 6-4  
start document parameters, defining, 6-4  
start page parameters, defining, 6-4  
Submit Job (P98305W), 4-1  
symbol set, order of precedence for PCL printing, 5-18  
system functions  
initializing logical printer name, 3-2  
using K2DoInitPrinter, 3-2  
system hierarchy  
using to resolve priority conflicts, 6-22

## T

---

temp directory, locating PDF files run on the web client, 5-22  
text out parameters, defining, 6-4

## U

---

UNIX, adding printers, 5-3

## W

---

web client  
locating log files, 5-22  
locating PDF files, 5-22  
modifying the jas.ini, 5-23  
Work With Bar Code Font form, 5-16  
Work With Batch Versions - Available Versions form, 5-13  
Work With Servers (P986116), 4-4

