

PeopleSoft®

EnterpriseOne 8.93
Connectors
PeopleBook

May 2004

EnterpriseOne 8.93
Connectors PeopleBook
SKU TT893C0504

Copyright© 2004 PeopleSoft, Inc. All rights reserved.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. ("PeopleSoft"), protected by copyright laws and subject to the nondisclosure provisions of the applicable PeopleSoft agreement. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft.

This documentation is subject to change without notice, and PeopleSoft does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft in writing.

The copyrighted software that accompanies this document is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this document, including the disclosure thereof.

PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, PeopleTalk, and Vantive are registered trademarks, and Pure Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice.

Open Source Disclosure

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

PeopleSoft takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation.

Table of Contents

About These EnterpriseOne PeopleBooks Preface	1
EnterpriseOne Application Prerequisites	1
Obtaining Documentation Updates	1
Typographical Conventions and Visual Cues	2
Typographical Conventions.....	2
Visual Cues	3
Comments and Suggestions	3
Connectors	4
Choosing the Connector Solution	5
COM Interoperability Solution	6
PeopleSoft EnterpriseOne and COM	6
COM Objects.....	6
COM Interoperability Usage.....	7
PeopleSoft EnterpriseOne COM Server	8
COM Connector	9
Generated COM Components	9
GenCOM	10
Setting Up an Environment for GenCOM	11
Example: Include Directories	11
Example: Lib Directories	12
Example: MSDev Directories	12
Example: Paths	12
Using the COM Generator (GenCOM).....	13
Running GenCOM.....	13
ProgID	15
Using GenCOM Output	16
COM Server Deployment	19
Installing COM Connector on a Non-PeopleSoft EnterpriseOne Client Environment.....	20
Setting Up a DCOM Server	22
Using the COM Wrapper Version Checker (CheckVer)	30
Running CheckVer	30
Using the IJDETimeZone Interface	31
XML File generated by GenCOM for IJDETimeZone	31
Using BHVRCOM via COM.....	32
Inbound XML Request via COM Server	33
COM Interoperability Transactions.....	33

COM+ Two-Phase Commit Transaction	34
Setting Up the COM+ Environment.....	34
Running the COM+ Two-Phase Commit Sales Order Entry.....	35
Case 1: PeopleSoft EnterpriseOne Participates in COM+ Transaction.....	39
Case 2: PeopleSoft EnterpriseOne Participates in a Distributed Transaction.....	43
OCM Support for the COM Connector	46
Installation Information	48
Using COM Tracing and Logging	48
Troubleshooting	48
COM Reliability	49
COM Connector Events	50
Registering Components.....	50
Event Subscriptions	53
Logging COM Events	54
Implementing the PeopleSoft EnterpriseOne Interface	54
Java Interoperability Solution	69
Dynamic Java Connector and PeopleSoft EnterpriseOne	71
Design Time for the Dynamic Java Connector	71
Runtime for the Dynamic Java Connector	82
Sample Applications.....	88
Java Connector and PeopleSoft EnterpriseOne	90
JDEDate.....	92
JDEMathNumeric	93
Design Time for the Java Connector	93
Setting Up a Client Environment for GenJava	95
Installing a Java Connector.....	96
Runtime for the Java Connector	97
Java Connector Outbound Events	113
Developing the Java Client to Use the Java Connector Outbound Event	
Source	114
J2EE Connector Architecture Resource Adapter	118
JCA 1.0 Specification Optional Features	119
Assembly and Components	121
Deployment and Configuration.....	122
Common Client Interface	125
Samples	129
Troubleshooting Checklist.....	133
jdeinterop.ini	134
[OCM]	134
[CACHE].....	135
[JDENET]	135
[SERVER]	135
[SECURITY]	136
[DEBUG]	136
[INTEROP]	137

[EVENTS]	138
[JDBj-BOOTSTRAP SESSION]	138
[JDBj-BOOTSTRAP DATA SOURCE]	139
[JDBj-CONNECTION POOL]	140
[JDBj-JDBC DRIVERS]	141
[JDBj-ORACLE].....	141
jdelog.properties	142
iJDEScript	144
iJDEScript Commands	145
Build Command.....	145
Call Command.....	145
Define Command	145
Define! Command	146
Exit Command.....	146
Help Command	146
Import Command	146
Importlib Command.....	147
Interface Command.....	147
Library Command.....	147
Login Command	148
Logout Command.....	148
Opt Command	148
Rename Command	149
Say Command	149
Sub Command	149
System Command.....	150
EnterpriseOne PeopleBooks Glossary	151
Index	187

About These EnterpriseOne PeopleBooks

Preface

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

This preface discusses:

- EnterpriseOne application prerequisites
- Obtaining documentation updates
- Typographical elements and visual cues
- Comments and suggestions

Note

EnterpriseOne PeopleBooks document only fields that require additional explanation. If a field is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line.

EnterpriseOne Application Prerequisites

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

See the *Foundation Guide*.

You might also want to complete at least one EnterpriseOne introductory training course.

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

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

Obtaining Documentation Updates

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

Note

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

See Also

PeopleSoft Customer Connection Website, <http://www.peoplesoft.com/corp/en/login.jsp>

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions
- Visual cues

Typographical Conventions

The following table contains the typographical conventions that are used in EnterpriseOne PeopleBooks:

Typographical Convention or Visual Cue	Description
<i>Italics</i>	Indicates emphasis, topic titles, and titles of PeopleSoft or other book-length publications. Also used in code to indicate variable values.
Key+Key	A plus sign (+) between keys means that you must hold down the first key while you press the second key. For example, Alt+W means hold down the Alt key while you press W.
Monospace font	Indicates a PeopleCode program or other code example.
“ ” (quotation marks)	Indicates an adjective that is used in a way that might not be readily understood without the quotation marks, for example "as of" date, "as if" currency, "from" date, and "thru" date.
Cross-references	EnterpriseOne PeopleBooks provide cross-references either below the heading "See Also" or preceded by the word See. Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Visual Cues

EnterpriseOne PeopleBooks contain the following visual cues:

- Notes
- Cautions

Notes

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

Note

Example of a note.

Cautions

Text that is preceded by *Caution* is crucial and includes information that concerns what you must do for the system to function properly.

Caution

Example of a caution.

Comments and Suggestions

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

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

Or you can send e-mail comments to doc@peoplesoft.com.

While we cannot guarantee an answer to every e-mail message, we will pay careful attention to your comments and suggestions.

Connectors

Connectors are point-to-point component-based models that allow third-party applications and PeopleSoft EnterpriseOne to share logic and data. The EnterpriseOne connector architecture includes Java and Component Object Model (COM) connectors and provides the following:

- Access to business functions
- Session management
- Point of entry
- Connection pooling
- Inbound transaction functionality
- Outbound event functionality

Using connectors provides additional benefits, such as:

- Connectors are scalable.
- Connectors provide multi-threading.
- Connectors allow concurrent users.

The following connectors are supported:

- Java connector – Java is a portable language, so you can easily tie EnterpriseOne functionality to Java applications.
- Dynamic Java connector – Provides the same type of functionality as the Java connector, but does not require you to generate business function wrappers.
- COM connector – The COM connector solution is fully compliant with the Microsoft Component Object Model. You can easily tie EnterpriseOne functionality to Visual Basic and VC++ applications.

The PeopleSoft connectors can receive and send XML documents. The connector architecture provides the capability to expose C and Java APIs for XML documents. Some of the benefits of using XML documents are listed below:

- You can use XML documents to aggregate business function calls into one object, which reduces network traffic.
- Because XML processing is based on the connector architecture, XML processing is scalable and multiple connections can be opened.
- XML processing supports XML CallObject, XMLList, and XMLTrans.

Choosing the Connector Solution

Do the following to decide which connector best satisfies your needs:

- Identify the logic or data that you want to access in PeopleSoft EnterpriseOne.
- Decide whether you want to use business functions exposed through a connector directly or XML documents.

Then decide whether to use a COM connector or a Java connector. If you are using an application server, the following guidelines can help you decide which connector to choose:

- If you are using Site Server, Commerce Server, or .NET, consider the COM connector.
- If you are using J2EE-based application server, consider the Java connector.
- The Java connector supports Java Connector Architecture Resource Adapter (JCA).

After you decide which connector to use, perform the following tasks:

1. Generate business function wrappers, if necessary. The dynamic Java connector does not require generating business function wrappers.
2. Install the interoperability server (connector).
3. Connect your application to the connector.

Sections in this guide explain how to perform the above tasks.

See Also

- ❑ *World Wide Web Consortium (W3C) XML home page* <http://www.w3.org/XML/> for general information about XML

The following topics in the *Interoperability Guide*:

- ❑ *Interoperability Overview* for information about the PeopleSoft EnterpriseOne models and capabilities that support these models
- ❑ *Business Function Calls* for information about choosing which business function to use
- ❑ *XML* for information about XML CallObject, XML Transaction, and XML List features
- ❑ *Events* for information about Z events, real-time events, and XAPI events

COM Interoperability Solution

COM allows developers to build systems by assembling reusable components from different vendors. COM provides logic and data sharing among disparate applications. COM is a binary interoperability specification and communication convention for software components. It is a single-vendor technology that is available on Microsoft platforms only. Since most independent software components are also self-contained, they are frequently called objects or servers.

Being a binary specification, COM is inherently independent of programming language. Unlike software libraries or DLLs, which are compiled to specific language or linkage conventions, COM-based software components are created ready to work with any COM client. For example, a Visual C++ application can use COM objects created in Visual Basic, or a VBScript within an intranet web page to control a COM object written in MicroFocus COBOL.

The COM connector provides a mechanism for executing business functions on the PeopleSoft EnterpriseOne server. You use the GenCOM utility on your Windows client to generate wrappers for objects. The wrappers can be deployed on any machine. You can develop application code for the generated wrappers using Visual Basic (VB) or C++. Once the objects change in the package, the connector communicates with the EnterpriseOne server for login, logoff, transactions, and for each business function execution call. The COM connector also supports subscribe and publish functionality for EnterpriseOne events.

Distributed Component Object Model (DCOM) enables COM objects in a distributed environment.

You can use COM+ transactions, which enable COM applications and third-party applications to take part in distributed transactions.

PeopleSoft EnterpriseOne and COM

Using COM, PeopleSoft EnterpriseOne exposes all master and major business functions through the interface definition language (IDL) standard. With COM, EnterpriseOne can pass logic and data requests to other applications via COM wrappers. These wrappers provide common interoperability methods across dissimilar systems. A wrapper is attached to each master and major business function and provides stubs for third-party applications to access.

COM Objects

A business function is a logical collection of C functions and their associated data structures grouped together to produce a unit of work. COM objects are wrappers around these business functions and data structures.

The interface provided by the COM wrappers has a one-to-one correspondence with the business functions. For example, if within the system library a business function named B550001 exists, and within this business function two C functions, named foo1 and foo2 exist with data structures for each function, named DS1 and DS2, the corresponding COM object would be:

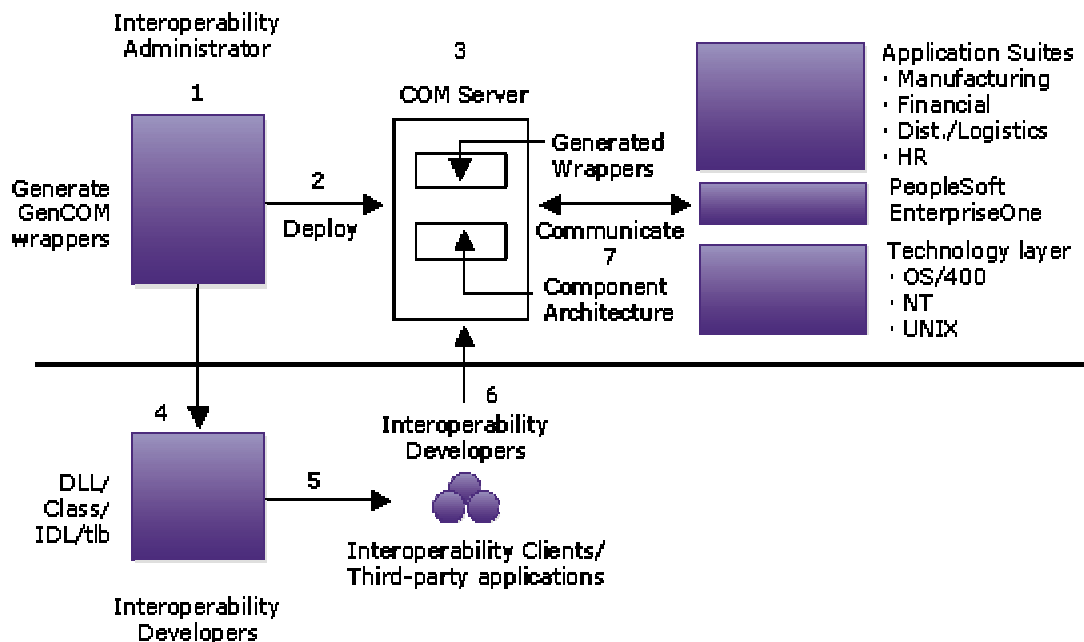
```

Interface IDS1
{
...
}
Interface IDS2
{
...
}
Interface IB550001
{
    HRESULT foo1 (IDS1 * param, IConnector* conn, long accessNumber);
    HRESULT foo2 (IDS2 * param, IConnector* conn, long accessNumber);
}
Their associated program IDs (ProgID) would be:
IDS1    - □jdeDS1.jdeDS1.1"
IDS2    - □jdeDS2.jdeDS2.1"
IB550001 - □jdeB550001.jdeB550001.1"

```

COM Interoperability Usage

The following illustration shows how the COM interoperability solution typically flows:



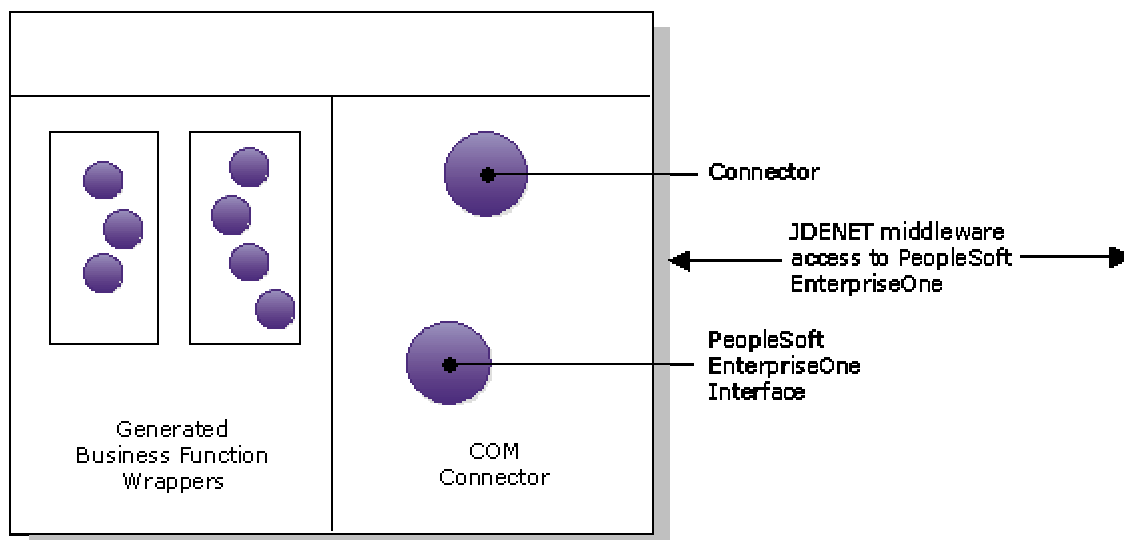
1. The administrator generates the COM wrappers.
2. The administrator deploys the COM objects to the COM server.
3. The COM server allows communication with the application server so that the generated COM objects can be used in applications.
4. Once the COM objects are on the COM server, the COM objects are configured to communicate with the application server.
5. The DLLs or IDLs from the generated COM objects are copied so that developers can use them.
6. Application developers create the applications.
7. The applications communicate with the COM server.

PeopleSoft EnterpriseOne COM Server

The PeopleSoft EnterpriseOne COM server contains two parts:

- Generated EnterpriseOne COM components (wrappers)
- COM connector

The following diagram shows the two parts of the COM server:



COM Connector

The COM server provides an interface to PeopleSoft EnterpriseOne, executes business functions within valid transactions, and provides error processing for interoperability clients. The main component of the COM server is the COM connector. The COM connector provides COM components that interface with EnterpriseOne and hosts the business component DLL generated by the GenCOM tool. The COM connector also provides the connector component that allows an interoperability client to log in and log out from EnterpriseOne. It manages all user sessions connected to the COM server. The following binaries combine to comprise the COM connector:

Binary	Explanation
JDECOMConnector2.exe	Primary interface for login and createBusinessObjects. Also maintains the users and business objects created.
JDECOMMN.dll	Interface for JDEMathNumeric and JDETimeZone.
Callobject.dll	Internal to JDECOMConnector.exe.
Comlog.dll	Used for logging, cache, and OCM lookup.
EventClass.dll	PeopleSoft EnterpriseOne event class that is implemented to receive events.
EventListener.dll	Receives events from the PeopleSoft EnterpriseOne server and publishes the events to COM+ Events.
EventManager.dll	Provides the interface for subscribe, unsubscribe, getList, and getTemplate for events.
jdeunicode.dll	The Unicode library, which is internal to PeopleSoft EnterpriseOne.
OneWorldInterfaceTx.dll	Provides the interface for PeopleSoft EnterpriseOne transactions and COM+ two-phase commit transactions.
Xmlinterop.dll	Contains the JDENET transport mechanism and the XMLRequest.

The JDECOMConnector2.idl defines the COM interfaces of the COM connector. JDECOMConnector2.idl is available under the Include directory.

The COM connector is available with the PeopleSoft EnterpriseOne server and client install.

Generated COM Components

GenCOM is included in the client installation. GenCOM uses an iJDEScript file as input to generate a COM DLL that is hosted by the COM connector. The iJDEScript file specifies wrapper components for business functions. Once the generated wrapper components are registered to the COM environment, they can be used to access business function functionality.

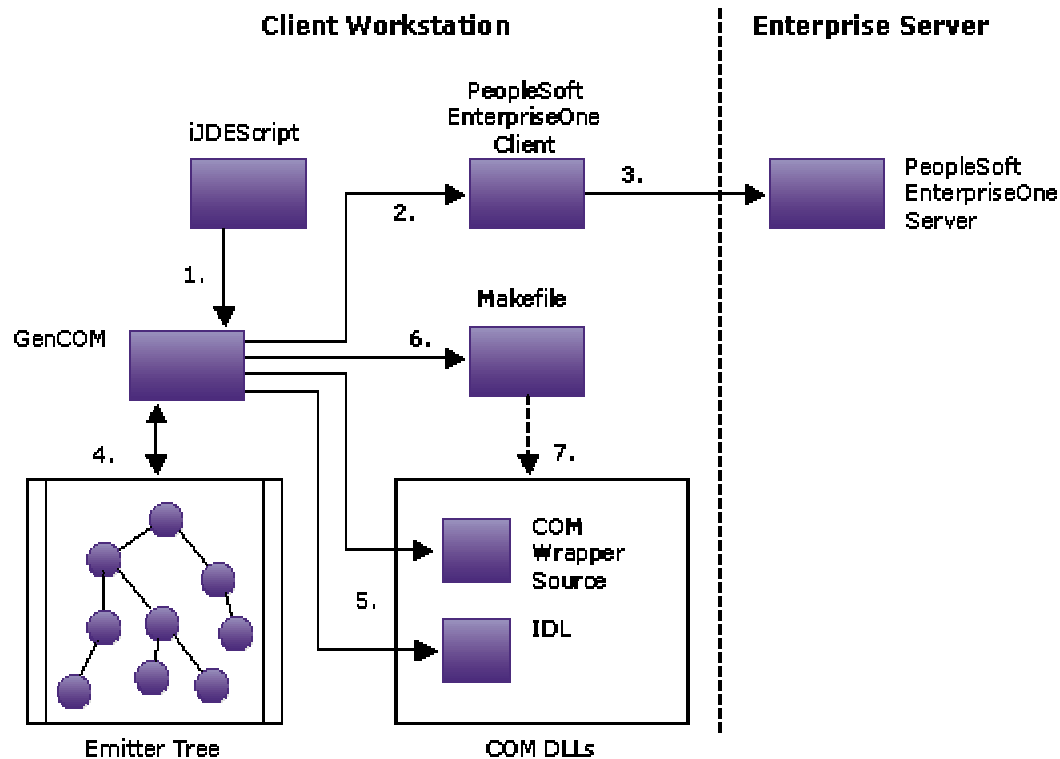
See Also

- ❑ *iJDEScript* in the *Connectors Guide* for more information about iJDEScript

GenCOM

GenCOM is a client tool that generates PeopleSoft EnterpriseOne COM components. GenCOM is a command line tool that reads a script file to determine which components to generate. GenCOM uses a multi-pass process to generate COM components.

The following illustration shows the process:



1. GenCOM reads the iJDEScript file.
2. GenCOM fetches the metadata for the business functions specified in the iJDEScript file.
3. GenCOM resolves dependency on the data structure.
4. GenCOM creates an internal emitter tree for the library to be generated.
5. GenCOM reads each node of the internal emitter tree and generates the appropriate COM code.
6. GenCOM generates a make file.
7. GenCOM compiles and builds the COM DLL from the generated code.

Setting Up an Environment for GenCOM

Setting up an NT client environment involves several steps. You should make sure that the following items are set up appropriately:

- Include directories
- Lib directories
- MSDev directories
- Paths

Example: Include Directories

< Directory where Microsoft SDK files are located>\include

Example: C:\Program Files\Microsoft SDK\include

< Directory where Microsoft program files are located>\VC98\atl\include

Example: C:\Program Files\Microsoft Visual Studio\VC98\atl\include

< Directory where Microsoft program files are located>\VC98\mf\include

Example: C:\Program Files\Microsoft Visual Studio\VC98\mf\include

< Directory where Microsoft program files are located>\VC98\include

Example: C:\Program Files\Microsoft Visual Studio\VC98\include

< Directory where PeopleSoft EnterpriseOne is located and release either Master, Prod, or Pristine>\include

Example 1: D:\B9\MSTB9\include

Example 2: D:\B9\PROD\include

< Directory where PeopleSoft EnterpriseOne is located and release either Master, Prod, or Pristine>\includeV

Example: D:\B9\SYSTEM\includeV

< Directory where PeopleSoft EnterpriseOne is located and release either Master, Prod, or Pristine>\include

Example: D:\B9\SYSTEM\include

Example: Lib Directories

< Directory where Microsoft SDK files are located >\lib

Example: C:\Program Files\Microsoft SDK\lib

< Directory where Microsoft program files are located >\VC98\mfc\lib

Example: C:\Program Files\Microsoft Visual Studio\VC98\mfc\lib

< Directory where Microsoft program files are located >\VC98\lib

Example: C:\Program Files\Microsoft Visual Studio\VC98\lib

< Directory where Microsoft program files are located >\Common\MSDev98\Bin

Example: C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin

< Directory where PeopleSoft EnterpriseOne is located >\System\Lib32

Example: D:\B9\System\Lib32

Example: MSDev Directories

< Directory where Microsoft program files are located >\Common\MSDev98

Example: C:\Program Files\Microsoft Visual Studio\Common\MSDev98

< Directory where Microsoft DevStudio is located >\SharedIDE

Example: C:\Program Files\DevStudio\SharedIDE

Example: Paths

< Directory where Microsoft SDK files are located >\bin

Example: C:\Program Files\Microsoft SDK\bin

< Directory where Windows NT is located >\System32

Example: C:\Winnt\System32

< Directory where Microsoft program files are located >\Common\Tools\Winnt

Example: C:\Program Files\Microsoft Visual Studio\Common\Tools\Winnt

< Directory where Microsoft program files are located >\Common\Msdev98\Bin

Example: C:\Program Files\Microsoft Visual Studio\Common\Msdev98\Bin

< Directory where Microsoft program files are located >\Common\Tools

Example: C:\Program Files\Microsoft Visual Studio\Common\Tools

< Directory where Microsoft program files are located >\Vc98\Bin

Example: C:\Program Files\Microsoft Visual Studio\Vc98\Bin

< Directory where Microsoft DevStudio is located>\SharedIDE\Bin\Ide

Example: C:\Program Files\DevStudio\SharedIDE\Bin\Ide

< Directory where Microsoft DevStudio is located>\SharedIDE\Bin

Example: C:\Program Files\DevStudio\SharedIDE\Bin

< Directory where PeopleSoft EnterpriseOne is located>\System\Bin32

Example: D:\B9\System\Bin32

In an NT environment, binaries are not compatible between the client and server machine. Do not copy .dll files or .exe files compiled on an NT workstation to an NT server. The struct alignments required by the EnterpriseOne server and the EnterpriseOne client are different.

Using the COM Generator (GenCOM)

You can run GenCOM to expose objects through COM. In a development environment, developers may run the COM Generation tool. In a production environment, a system administrator should run the COM Generation Tool.

When you use GenCOM, use the iJDEScript scripting language to script code generation activities.

Running GenCOM

You run GenCOM from the command line. Several options are available for generation. The COM Generation Tool is in <install>\system\bin32\GenCOM.exe.

Syntax

GenCOM [options] [libraries]

Options

/? Lists the options available for generation.

/C++ <option> Provides GenCOM with the compiler options you want to use in the generation of the COM servers.

/Cat <category>	Tells GenCOM to generate wrappers based on the following categories: • master business functions • major business functions • minor business functions • uncategorized business functions
/CL <file>	Tells GenCOM what compiler (.exe) to use for compilation.
/Cmd *	Processes code generation commands from the console.
/Cmd <filename>	Processes code generation commands from <filename>.
/Debug	Builds debug information (.pdb and .bsc files) into the libraries so that the Visual Studio debugger can access source information.
/EnvironmentID <env>	Provides GenCOM with the environment in which you want to sign in to PeopleSoft EnterpriseOne.
/ErrFile <file>	Provides GenCOM with the filename to log errors produced by GenCOM during the generation process, for example, "errors.log."
/MIDL	Provides GenCOM with the MIDL compiler options you wish to use in the generation of the COM servers.
/MTL <file>	Tells GenCOM which MIDL compiler (.exe) to use for compilation.
/ListLibraries	Lists all the available libraries that you can run GenCOM against.
/MsgFile <file>	Provides GenCOM with the filename to log messages produced by GenCOM during the generation process, for example, "messages.log."
/NoBSFN	Tells GenCOM not to create wrappers for business functions. This option is for generating parameter sets only.
/NoCompile	Tells GenCOM to only generate the source files without compiling.
/NoDebug	Optimizes libraries for space using the /O1 Visual C++ compiler option.
/Out <path>	Provides GenCOM with the directory path in which to place the output files, for example "C:\winnt\system32."
/OWRelease flag for GenCOM	You can override the OWRelease information by activating this flag and typing a string that specifies the version information. PeopleSoft recommends that you follow a naming convention that is consistent throughout the implementation or use the default version information that is generated by GenCOM.
/Password <password>	Provides GenCOM with the password with which you want to sign in to EnterpriseOne.

/Role	Provides GenCOM with the role with which you want to sign in to EnterpriseOne.
/STA	Generates STA components. (By default, all generated components are MTA and are optimized for scalability and performance. /STA allows you to generate STA components if you need them.)
/TempOut <path>	Provides GenCOM with the directory path in which to place temporary files needed for the build process, for example, "C:\temp."
/UserID <userid>	Provides GenCOM with the user name with which you wish to sign in to PeopleSoft EnterpriseOne.

For example, if you want to see available libraries that you can run GenCOM against, you would enter the following command from your system command line:

```
C:\B9\System\Bin32>gencom /ListLibraries
```

Running GenCom Entry

The following command generates COM wrappers for Category 1 business functions in the CAEC library:

```
GenCOM /Cat 1 /UserID Devuser1 /Password Devuser1 /Environment ADEVHP02
CAEC
```

ProgID

Each time GenCOM generates a wrapper, it creates a ProgID for each COM component. The ProgID identifies the COM component in the registry. The ProgID is independent of PeopleSoft EnterpriseOne and is based on the library and the interface specifications in the script file. The key, OneWorldRelease, contains the EnterpriseOne release and environment information. For example, if the library name is AddressBook and the interface name is JDESalesOrderEntry, then the ProgID will be AddressBook.JDE AddressBook. If GenCOM is run with environment DV9NIS2, then the OneWorldRelease key will contain DV9NIS2. If a type mismatch exists, you will receive a warning.

The CompatibleEnvironment key remembers the list of EnterpriseOne environments with which the wrapper is compatible. If an environment is not on the list or is listed as incompatible, the COM client receives an error message when trying to create the object with the environment.

The following ProgID illustrates the standard ProgID naming conventions:

```
HKEY_CLASSES_ROOT\
CLSID\{77454442-7941-44BB-9BCB-4253E80AC8B3}
\InprocServer32 C:\B9\System\IDA\Samples\AddressBook\AddressBook.dll
\ProgID SalesOrderEntry.JDESalesOrderEntry
\VersionIndependentProgID AddressBook.JDEAddressBook
\OneWorldRelease DV9NIS2
\CompatibleEnvironment DV9NIS2
```

Using GenCOM Output

The output for GenCOM produces fully functional COM servers based on the library to which you generate wrappers. Because you are interacting with the PeopleSoft EnterpriseOne system, you must follow security and installation procedures to gain access to the system.

You must have a fully licensed copy of PeopleSoft EnterpriseOne properly installed on the target machine. You must also sign in to the PeopleSoft EnterpriseOne environment. For the sign-in process, you use the jdeCOMConnector interface.

Visual Basic

The following code example demonstrates how to use a generated COM business function wrapper in Visual Basic. This example creates business objects. Refer to the AddressBook sample included with the COM interoperability software for a complete working example of this functionality.

```
Dim WithEvents OW As OneWorldInterface '//OneWorldInterface
Dim conn As New Connector             '//COM Connector
Dim AB as JDEAddressBook              '//AddressBook
Dim phone as D0100032                 '//Data Source
Dim Mailing As D0100031                '//Data Source
Dim AddressAs D0100033                 '//Data Source
Dim EffectiveDate As D0100019 '//Data Source
DimParentAddress As D0100381           '//Data Source
Dim sessionID As Long                 '//server Session ID
Private Sub Form_Load()
    sessionID=conn.Login("Foo", "Bar", "DV9NIS2")
    Set OW = conn.CreateBusinessObject("OneWorld.FunctionHelper.1", sessionID)
    Set AB = conn.CreateBusinessObject("AddressBook.JDEAddressBook", sessionID)
    Set phone = AB.CreateGetPhoneParameterset
    Set Mailing = AB.CreateGetMailingNameParameterset
    SetAddress = AB.CreateGetEffectiveAddressParameterset
    Set EffectiveDate = AB.CreateGetABEffectiveDateParameterset
    Set ParentAddress = AB.CreateGetParentAddressParameterset
End Sub
```

Visual C++

The following Visual C++ code example demonstrates how to create the connector and how to create a business function on the COM server. This example creates an AddressBook business function and uses GenCOM objects from C++.

```
#include <windows.h>
#include <stdio.h>
#include <objbase.h>
#include <comdef.h>
#include <wchar.h>

#include "addressbook.h"
#include "AddressBook_i.c"
#include "jdecomconnector2.h"
#include "jdecomconnector2_i.c"
#define IPhone ID0100032
#define IMailing ID0100031
#define IAddress ID0100033
#define IEffectiveDate ID0100019
#define IParentAddress ID0100381
#define SERVER OLESTR("COMSRV") //Change to the COM server.
#define ABNO 4242 //change this according to user input.
HRESULT CreateConnector( IConnector **ppConnector )
{
    HRESULT hr = E_FAIL;

    *ppConnector = 0;

    //NOTE: Pass a COSERVERINFO struct to activate on a remote machine
    COSERVERINFO csi = {0, SERVER, 0, 0};
    MULTI_QI mqi = { &IID_IConnector, 0, 0 };
    hr = CoCreateInstanceEx(CLSID_Connector, 0, CLSCTX_LOCAL_SERVER,
        0, // &csi,
        1, &mqi);

    if(SUCCEEDED(hr) && SUCCEEDED(mqi.hr))
    {
        ppConnector = reinterpret_cast<IConnector*>(mqi.pItf);
    }
    return hr;
}

HRESULT Login( IConnector **pConnector, IOneWorldInterface **ow, long *accessno )
{
    HRESULT hr;
    IDispatch *idsptch = 0;

    printf("Login started\n");
    bstr_t User(L "Foo "), Password(L"Bar "), Env("DV9NIS2");
    hr = (*pConnector)->Login(User,Password,Env,accessno );

    if( !SUCCEEDED(hr))
```

```

{
    printf( Login failed with hr = %x",hr);
    return E_FAIL;
}
_bstr_t bo("OneWorld_FunctionHelper.1");
hr=(*pConnector)->CreateBusinessObject(bo, *accessno, &idsptch );
if( !SUCCEEDED(hr) || (!ow))
{
    Printf("CreateBusinessObject(OneWorld.FunctionHelper.1) failed with hr %x",hr);
    return E_FAIL;
}
hr=idsptch->QueryInterface(IID_IOneWorldInterface, (void **)ow );
if(!SUCCEEDED(hr) || (!ow))

```

```

}
    Printf( QueryInterface for IOneWorldInterface failed with hr "%x",hr);
    return E_FAIL
}
printf("Login completed \n");
return S_OK;
}
HRESULT UseAddressBook(IConnector *pConnector, IOneWorldInterface *ow,
long*accessno)
{
    HRESULT hr;
    IJDEAddressBook *ab;

    IDispatch *idsptch;
    IPhone *phone;
    IMailing *Mailing;
    IAddress *Address;
    IEffectiveDate *EffectiveDate;
    IParentAddress ParentAddress;

    printf("Starting to use AddressBook\n");
    _bstr_t bo("AddressBook.JDEAddressBook");
    hr = pConnector->CreateBusinessObject(bo, *accessno, &idsptch);
    hr = idsptch->QueryInterface( IID_IJDEAddressBook, (void **)&ab);

    if(!SUCCEEDED(hr) || (tab))
    {
        printf( "CreateBusinessObject( AddressBook ) has failed with hr %x", hr);
        return E_FAIL;
    }
    return S_OK;
}

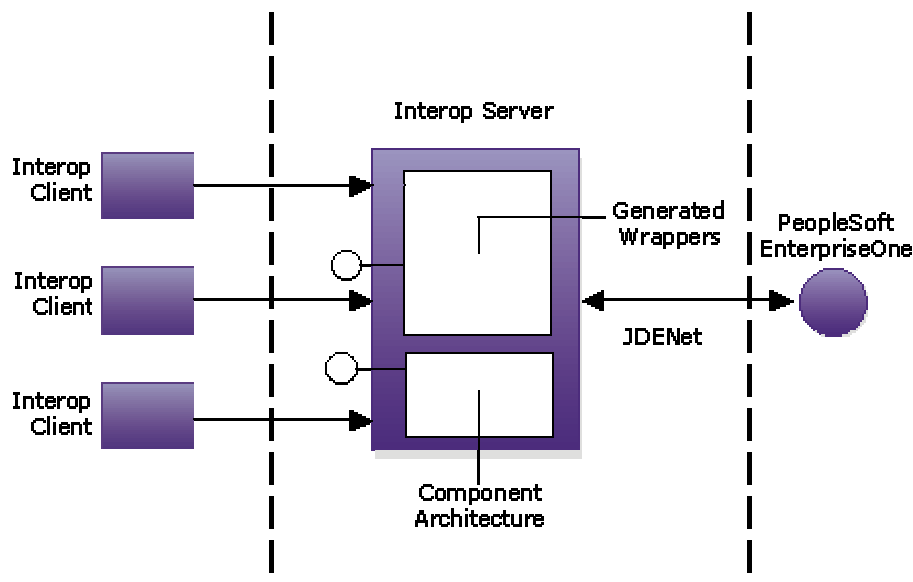
```

The following code creates the connector object and uses it to create a business function with its associated ParameterSet. The code then calls a method, Foo1, on the business object with the ParameterSet, the connector, and the access code returned by the act of logging on to the connector.

```
Int main(int argc, char *argv[])
{
    HRESULT hr;
    IOneWorldInterface *ow;
    long accessno;
    IConnector *pConnector;
    hr = CoInitializeEx(0, COINIT_MULTITHREADED);
    if(SUCCEEDED(hr))
    {
        hr = CreateConnector(&pConnector);
        if(SUCCEEDED(hr))
        {
            Login( &pConnector, &ow, &accessno );
            //Do more processing with AddressBook and logoff at the end.
        }
        CoUninitialize();
    }
}
```

COM Server Deployment

The COM server uses socket-based middleware to access the PeopleSoft EnterpriseOne application server. The jdeinterop.ini file must be configured to specify the PeopleSoft EnterpriseOne server. The COM server reads the jdeinterop.ini file and opens the socket connection to the specified application server.



Installing COM Connector on a Non-PeopleSoft EnterpriseOne Client Environment

Installing the COM connector on a non-PeopleSoft EnterpriseOne client machine requires the following steps:

1. Copy the following files from the EnterpriseOne server (system\bin32) to a directory on the desired machine. For example, copy the files in c:\program files\PeopleSoft to a non-PeopleSoft EnterpriseOne client machine.
 - JDECOMConnector2.exe
 - JDECOMMN.dll
 - callobject.dll
 - comlog.dll
 - EventManager.dll
 - OneWorldInterfaceTx.dll
 - xmlinterop.dll
 - jdel.dll
 - jdethread.dll
 - jdeunicode.dll
 - ustdio.dll
 - icuil8n.dll
 - jdeinterop.ini to c:\(root directory)
 - checkver.exe
 - ICUUC.dll
 - Icu\data*.*
 - IXXML4C2_3.dll
 - EventClass.dll
 - EventListener.dll
2. Create a new directory Icu\data\ on the machine where the COM server is located. Copy all of the files from the PeopleSoft EnterpriseOne server in folder system\Locale\xml*.* into Icu\data\ . Create a new system variable, *ICU_DATA*, in the environment variables of the system properties and specify the path to the Icu\data\ as the value.
3. Execute the following command on the target location to register the COM connector components:
`c:\programfiles\PeopleSoft\JDECOMConnector2.exe /RegServer`
4. Run GenCOM on a PeopleSoft EnterpriseOne client machine and copy the output DLL and the wrapper components (for example, wrapper.dll) to the following directory:
`c:\programfiles\PeopleSoft\wrapper.dll`

5. Execute the following command to register the COM wrapper components:

```
c:\programfiles\PeopleSoft\regsvr32wrapper.dll
```

6. Create the JDEinterop.ini file.

Set the PeopleSoft EnterpriseOne Server and port values to the PeopleSoft EnterpriseOne application server with which you want the COM server to communicate.

Your COM server is now ready.

To unregister the COM server, use the /unreserved option. For example:

```
c:\programfiles\PeopleSoft\JDECOMConnector2.exe /unreserved
```

To unregister the COM wrapper, use the /u option. For example:

```
c:\programfiles\PeopleSoft\regsvr32 /u wrapper.dll
```

See Also

- *jdeinterop.ini* in the *Connectors Guide* for more information about the JDEinterop.ini file

Setting Up a DCOM Server

You can set up a DCOM server on a PeopleSoft EnterpriseOne server machine. DCOM enables COM objects in a distributed environment. The following list identifies the tasks for setting up the DCOM server.

- Set up DCOM for a server environment.
- Set up security on the COM server.
- Set up the identity as interactive user.
- Set up DCOM for a client environment.

To ensure that the interoperability client works properly, you must perform the set up DCOM for a server environment and set up DCOM for a client environment tasks.

► To set up DCOM for a server environment

1. Run GenCOM on a PeopleSoft EnterpriseOne client machine, with the following options:
`gencom /out <path> /tempout <path> /cmd App.cmd`

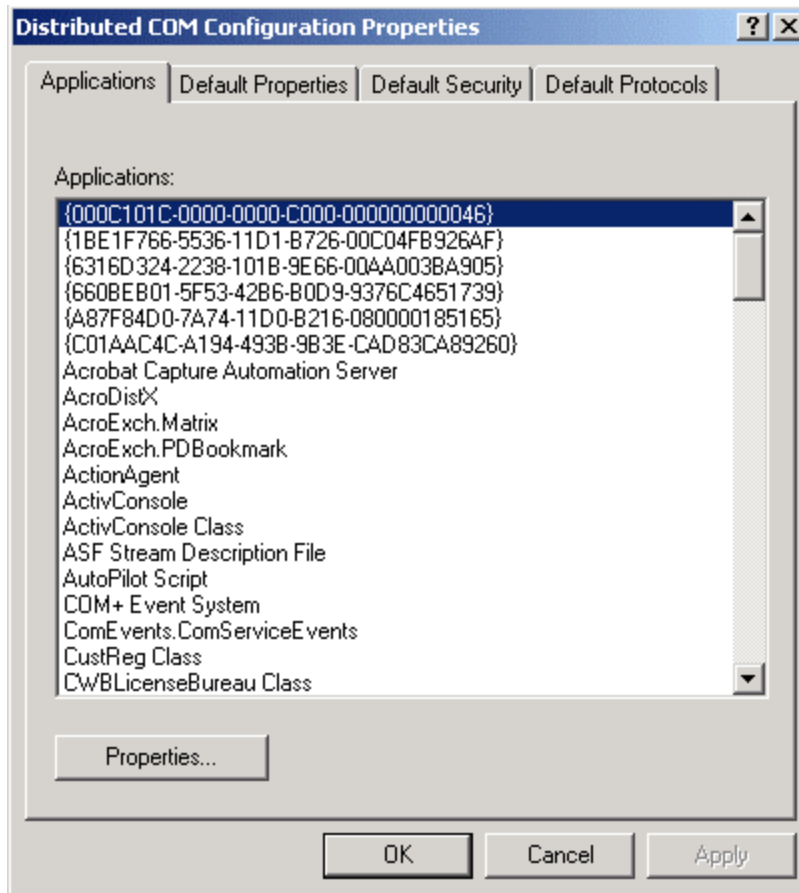
Because GenCOM is a PeopleSoft EnterpriseOne client-side only tool, you must perform this step on a PeopleSoft EnterpriseOne client machine.

2. Copy the App.DLL file and the App.TLB file generated by GenCOM to the COM server machine.
3. On the COM server machine, from the command line:
 - Run `jdecomconnector2.exe /RegServer`.
 - Run `regsvr32 App.dll`.
 - Set the correct security level for `jdecomconnector2.exe` and `App.dll`.

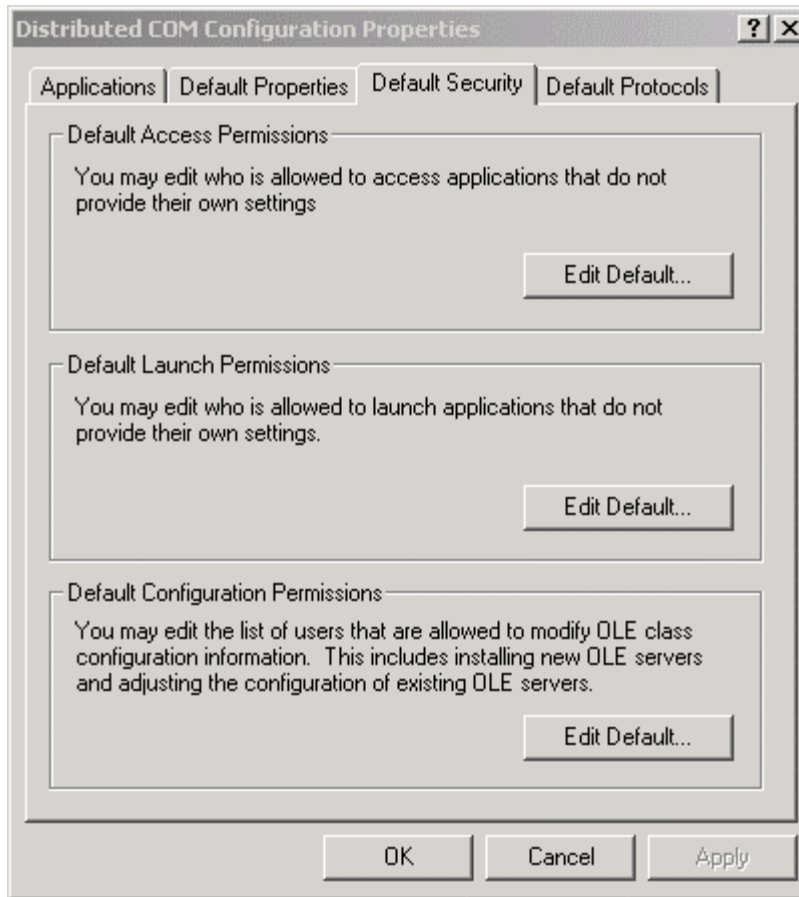
► To set up security on the COM server

1. From the Start menu, choose Run.
2. In the Open: field, enter `Dcomcnfg.exe`.

The Distributed COM Configuration Properties form appears.

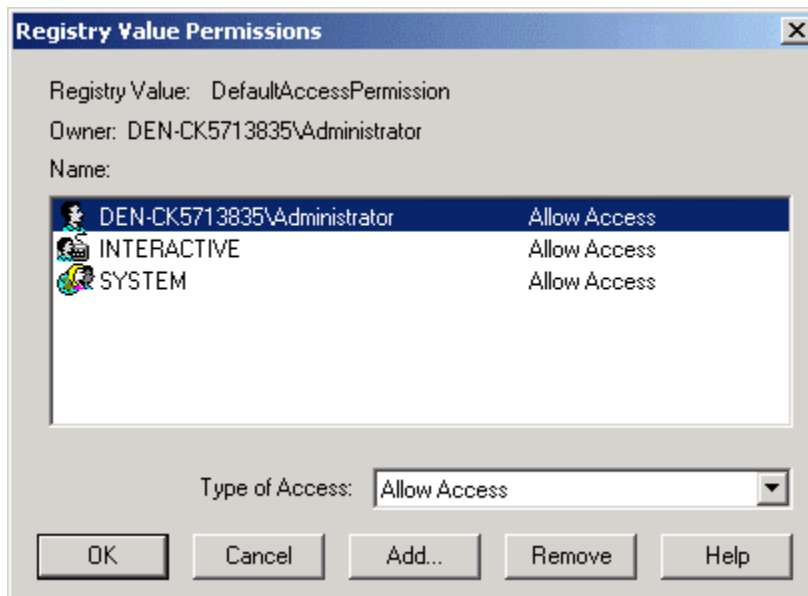


3. Click the Default Security tab.

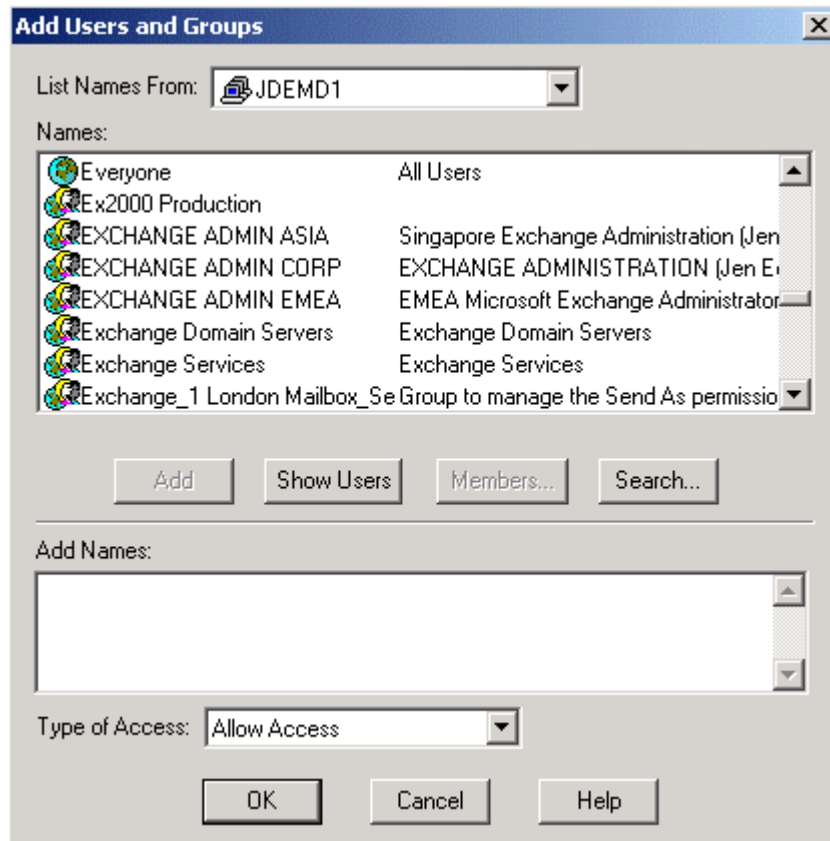


4. Click on the Edit Default Button in Default Access Permissions group.

The Registry Value Permissions form appears. Your computer might already have some entries present.



5. On Registry Value Permissions, click Add.



6. Click Everyone and click Add.

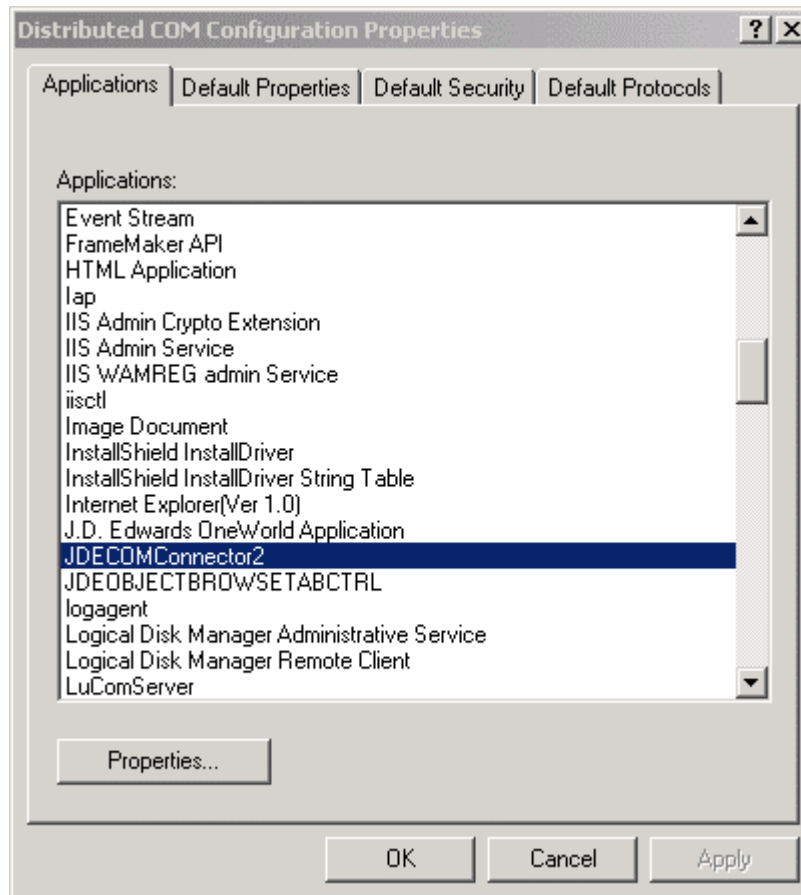
Type of Access should be Allow Access.

7. Click OK.

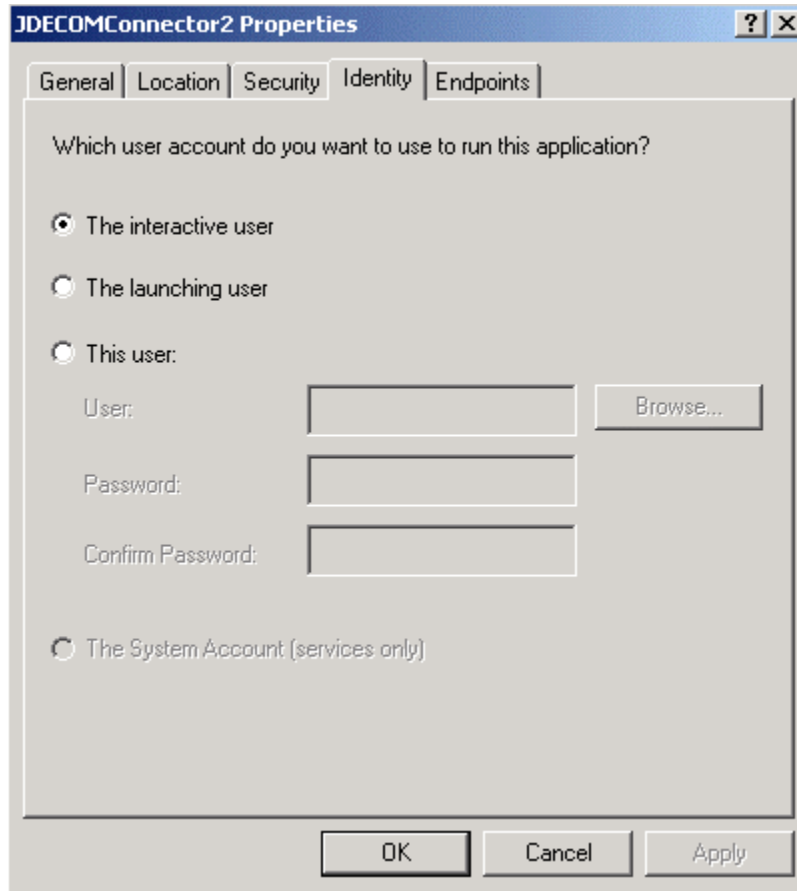
Repeat Steps 4 through 7 for Default Launch permissions. No setup is required for Default Configuration permissions.

► To set up the identity as interactive user

1. Run DCOMCnfg.



2. Click JDECOMConnector2 and click Properties.



3. Click the Identity tab and click the interactive user option.
4. Click Apply to apply the change.

Note

You must perform this step every time you register the connector. If you copy the JDECOMConnector2.exe using Explorer, Explorer will rerun the registration, and you must repeat the above steps.

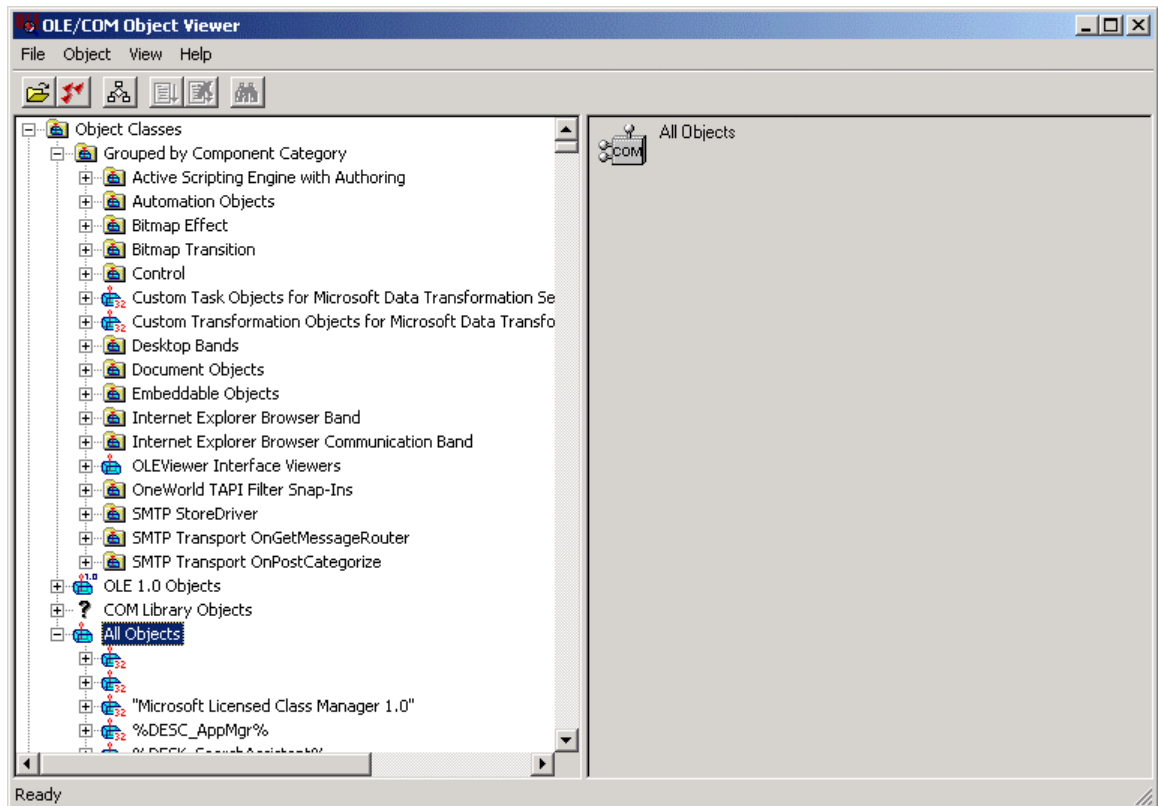
To use Callbacks (Connection Points) with the COM solution, repeat the same procedure on the COM client machine. Most of the shipped examples use Callbacks and require that you open the security on the client machine.

► To set up DCOM for a client environment

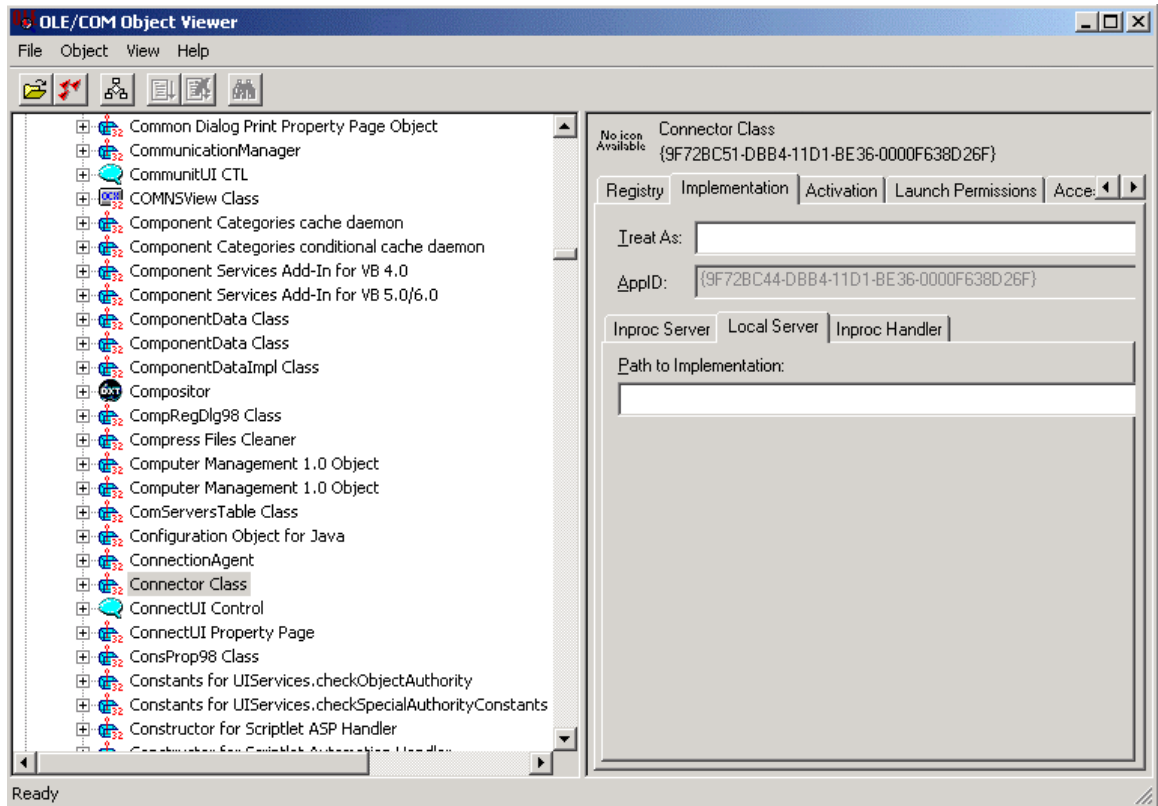
From a DOS prompt on the DCOM client machine, run jdecomconnector2.exe /RegServer:

1. Enter the following at the prompt:
oleview.exe
2. From the menu bar, choose oleview.
3. Click View and choose Expert Mode.

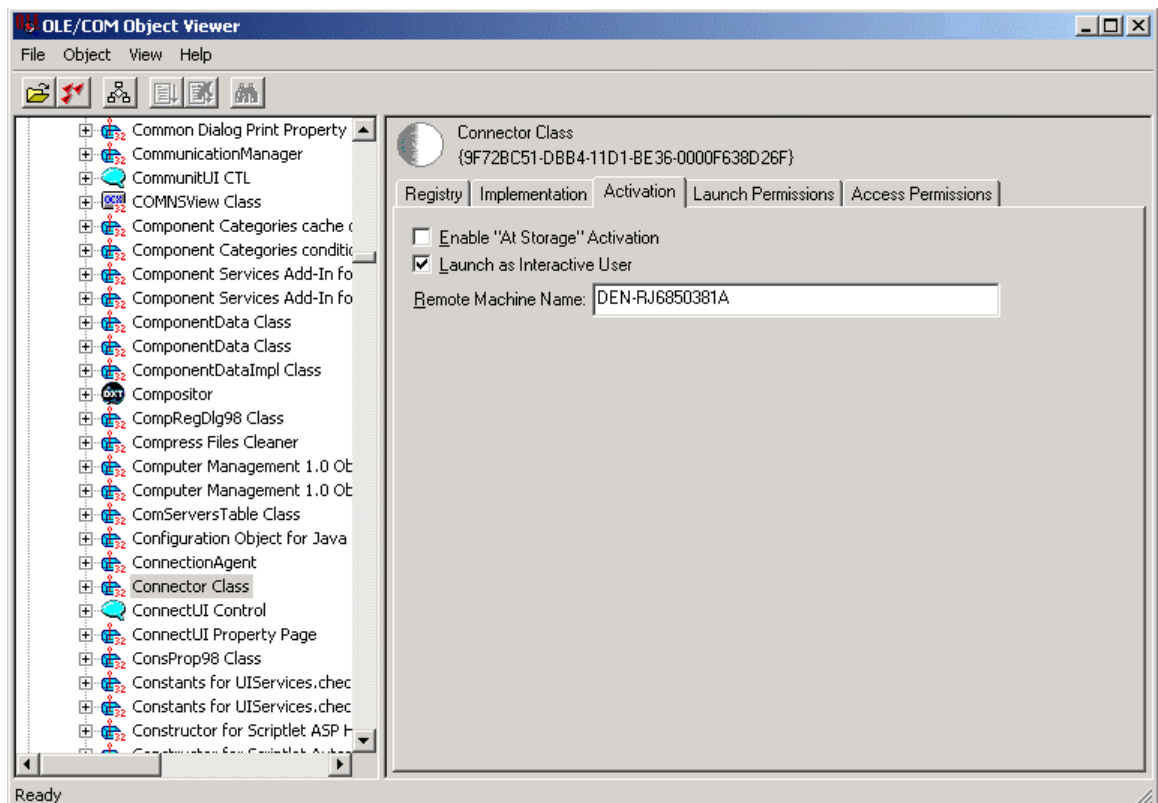
4. In the oview window under Object Classes, double-click All Objects, and wait for all objects to appear.



5. Under All Objects, find and click Connector Class.
6. Click the Implementation tab on the right side panel, and then click the local server and remove anything that appears in the editing window.



7. On the Activation tab, turn on the Launch as Interactive User option.
8. In Remote Machine Name, enter the COM server machine name.



9. Repeat steps 5 through 8 for MathNumeric Class.
10. Start the DCOM client application.

Remember

PeopleSoft EnterpriseOne client-only business functions are not reachable.

Using the COM Wrapper Version Checker (CheckVer)

You can run CheckVer to verify whether a previously generated COM object is compatible with another environment. Typically, a system administrator performs this task.

The XML files generated by GenCOM are the signatures of the objects generated against specific PeopleSoft EnterpriseOne environments. These XML files can be used with CheckVer to verify that the wrappers on the COM server are compatible with these environments.

When you introduce a new PeopleSoft EnterpriseOne environment, you run GenCOM against the new environment by using the /NoCompile option. You also use the iJDEScript that you used to generate the wrappers on the COM server to generate XML signature files for the objects in the new environment. Run CheckVer on the COM server with the newly generated XML files to verify that the new environment is compatible with wrappers on the COM server that was previously generated with a different environment. CheckVer updates the registry settings for the wrapper on the COM server according to the result of the compatibility test. If the new environment is incompatible, the COM client is not allowed to create business objects with the new environment.

Running CheckVer

CheckVer compares the XML signature file that is produced from GenCOM with the spec definitions on the local PeopleSoft EnterpriseOne client machine. You can run CheckVer from the command line on the COM server, or CheckVer can be run automatically as part of the GenCOM process.

To see the options that CheckVer provides, run the following command from the command line:

```
c:\>CheckVer.exe -?
```

Syntax

CheckVer [option] <filename>

Example

CheckVer -r addressbook.xml

Options

-r	CheckVer will only report whether the environment is compatible with the server but will not update the registry settings for the wrapper on the COM server with the result, and CheckVer will not validate the wrapper DLL.
-----------	--

Using the IJDETimeZone Interface

To modify and display the JDEUTIME data type in the appropriate format, the COM client and GenCOM must use the JDEUTIME APIs. Date and time information is displayed in a time based on the date and time that is in your personal profile or a time zone specified by an application.

The following steps, along with sample code, illustrate how to use the IJDETimeZone Interface.

1. Create the IJDETimeZone interface.

```
MULTI_QI mqi = { &IID_IJDETimeZone, 0, 0 };
hr = CoCreateInstanceEx(CLSID_IJDETimeZone, 0, CLSCTX_ALL, 0, 1, &mqi);
if (SUCCEEDED(hr) && SUCCEEDED(mqi.hr))
{
    *ppJdeTimeZone = reinterpret_cast<IJDETimeZone*>(mqi.pItf);
}.
```

2. Set the time for a time zone (UTC-5:30) for the data structure DXXXXXXX. If a time zone is not specified, then the time will be considered to be at UTC. If an invalid time zone string is passed, then an error occurs.

```
DATE dt;
BSTR bstrUTC = SysAllocString(L"UTC-5:30");
pJDETimeZone->put_DateTime(bstrUTC, &dt);
DXXXXXX->put_jdOrderDate(pJDETimeZone);
```

3. Get a time for a given time zone from PeopleSoft EnterpriseOne. If a time zone string is not passed, then the time and date stored in PeopleSoft EnterpriseOne, which is at UTC, will be returned. If an invalid time string is passed, then an error occurs.

```
DXXXXXX->get_jdOrderDate(pJDETimeZone);
DATE dt;
BSTR bstrUTC = SysAllocString(L"UTC-5:30");
pJDETimeZone->get_DateTime(bstrUTC, &dt);
```

XML File generated by GenCOM for IJDETimeZone

For each data item whose data type is JDEUTIME in the data structure DXXXXXXX, GenCOM generates an XML file, as follows:

```
<Signature environment="Environment Name">
  <Interface name="Interface Name">
    <Method name="BSFN">
      <Param name="DXXXXXXX" type="u" />
    </Method>
  </Interface>
</Signature>
```

Using BHVRCOM via COM

PeopleSoft EnterpriseOne clients use the BHVRCOM structure to control the execution of business functions. A COM client can use the IBHVRCOM interface to set and get BHVRCOM values for business functions. The interface definition is in the jdeconnector2.idl file.

The following Visual Basic code demonstrates how to query the IBHVRCOM interface and pass values to business functions:

```
Dim conn As New Connector ' // COM Connector
Dim WithEvents OW As OneWorldInterface ' // OneWorldInterface
Dim myBHVRCOM As IOneWorldBHVRCOM ' // BHVRCOM
Dim AB As JDEAddressBook ' // AddressBook
Dim phone As D0100032 ' // Data source
l = conn.Login("JDE", "JDE", "M7332RSO2")
Set OW = conn.CreateBusinessObject("OneWorld.FunctionHelper.1", l)
Set myBHVRCOM = OW ' // query the IOneWorldBHVRCOM interface
myBHVRCOM.iBobMode = 8 ' // set BHVRCOM values
myBHVRCOM.szApplication = "myApp"
myBHVRCOM.szVersion = "myVersion"
Set AB = conn.CreateBusinessObject("AddressBook.JDEAddressBook", l)
Set phone = AB.CreateGetPhoneParameterset
phone.mnAddressNumber = 1
AB.GetPhone phone, OW, conn, l ' // business function is executed with the BHVRCOM values
....
```

Some of the above code is explained in the following table:

Code	Explanation
myBHVRCOM.iBobMode=	BobMode is the mode (add, update, delete) of the interactive application. Values for BobMode are: • BOB_MODE_UNDEFINED = 0 • BOB_MODE_SPECIAL = 1 • BOB_MODE_ADD = 2 • BOB_MODE_ADD_PRIMARY = 3 • BOB_MODE_ADD_SPECIAL = 4 • BOB_MODE_DELETE = 5 • BOB_MODE_UPDATE = 6 • BOB_MODE_UPDATE_SPECIAL = 7 • BOB_MODE_INQUIRE = 8 • BOB_MODE_COPY = 9
myBHVRCOM.szApplication=	The value is the name of the interactive application.
MyBHVRCOM.szVersion=	The value is the version of the interactive application. This field can be used for localizations of the applications.

Inbound XML Request via COM Server

You can use the COM connector to send inbound synchronous XML requests (such as XML CallObject, XML List, and XML Transaction) to the PeopleSoft EnterpriseOne server. The COM connector uses XML APIs to access the inbound XML requests.

See Also

See the following topics in the *Interoperability Guide*:

- ❑ *XML CallObject* for more information about using XML CallObject
- ❑ *XML List* for more information about using XML List
- ❑ *XML Transaction* for more information about using XML Transaction

COM Interoperability Transactions

COM interoperability transactions include COM connector prepare, commit, and rollback functionality. The COM transaction interoperability solution supports the following types of transactions:

- Auto commit transactions
- Manual commit transactions

A transaction can be started as auto commit or manual commit. In auto commit, PeopleSoft EnterpriseOne automatically commits the transaction that has been started. If a transaction is started in manual commit, you have to explicitly call prepare and commit functionality for the transaction to be committed.

The COM connector also supports manual commit. Typically, a transaction is started in manual commit by calling BeginTransaction with the flag set to 1. Subsequent calls to prepare and commit commits the transaction. The COM connector prepare and commit does not support distributed transactions that involve transactions other than PeopleSoft EnterpriseOne.

Outline for Calling Prepare and Commit

<code>Dim soeOWInterface As OneWorldInterface</code>	Declare the OneWorldInterface.
<code>soeOWInterface.BeginTransaction (accessNumber, connector, txMode)</code>	Start the transaction in manual commit by calling begin transaction and setting the txMode to 1. 0 is for auto commit.
<code>//execute all BSFNs like the //enddoc and other BSFNs</code>	After a call to Begin Transaction is made, do all the transactions that you want to enclose within this manual commit before calling prepare.
<code>soeOWInterface.Prepare</code>	Call prepare when all of the transactions are done.

soeOWInterface.Commit (or) soeOWInterface.Rollback	Call Commit to commit the transaction (or) Rollback to roll back the transaction if an error occurs.
--	--

The default timeout value for a manual transaction is 5 minutes. If you do not commit the transaction within 5 minutes, the transaction context is freed and the transaction is rolled back. You can change the default timeout by setting the `manual_timeout` value in the [INTEROP] section of the `jdeinterop.ini` file. The value is in milliseconds.

COM+ Two-Phase Commit Transaction

The COM connector can participate in distributed transactions. The COM connector's ability to participate in distributed transactions enables any application that uses the COM connector to participate in the two-phase commit transaction. Applications that have the capability to participate in distributed transactions can also use the COM connector.

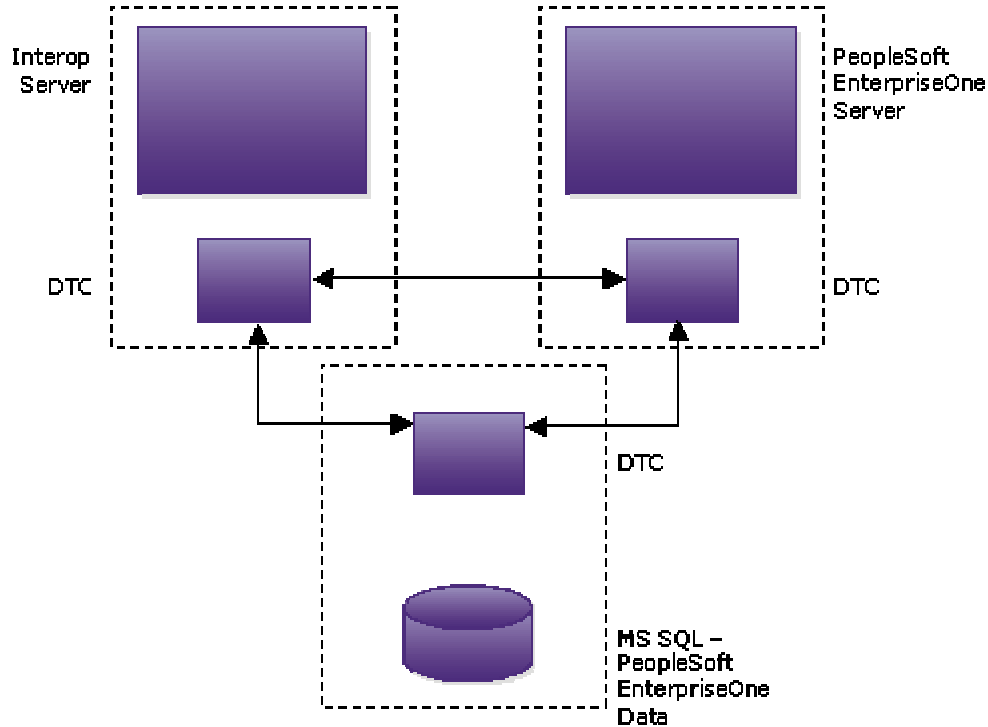
Setting Up the COM+ Environment

Typically, when you use COM+ for two-phase commit, you must set up the environment for the following three computers:

- COM connector
- PeopleSoft EnterpriseOne server
- Database server

A distributed transaction coordinator (DTC) is expected to run on each of the machines. Before testing the COM+ two-phase commit, you must make sure that the DTCs on each machine are correctly configured and that the DTCs talk to each other.

The following illustration shows the physical configuration:



Note

Typically, administrative rights are required for you to run the examples, which talk to DTCs on different machines. For more information about setting up DTC and various configurations, refer to the Microsoft documentation.

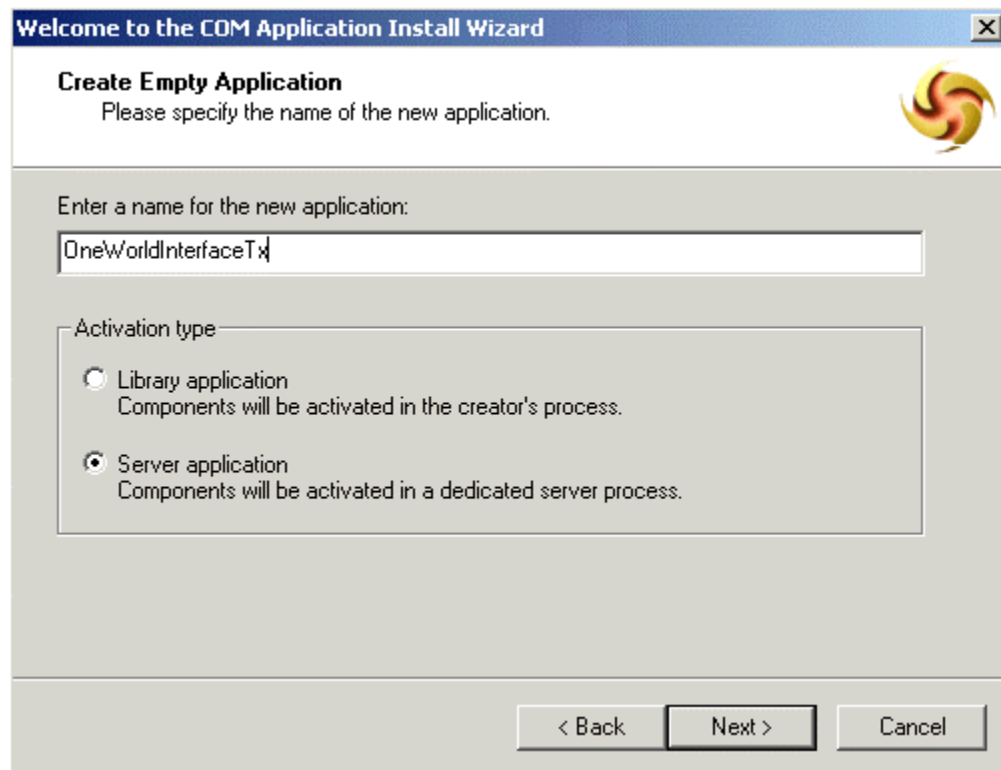
Running the COM+ Two-Phase Commit Sales Order Entry

A new COM+ dll (OneWorldinterfaceTx.dll) is provided to be used along with the COM connector to participate in a two-phase commit. OneWorldInterfaceTx.dll must be registered with the COM+ component services.

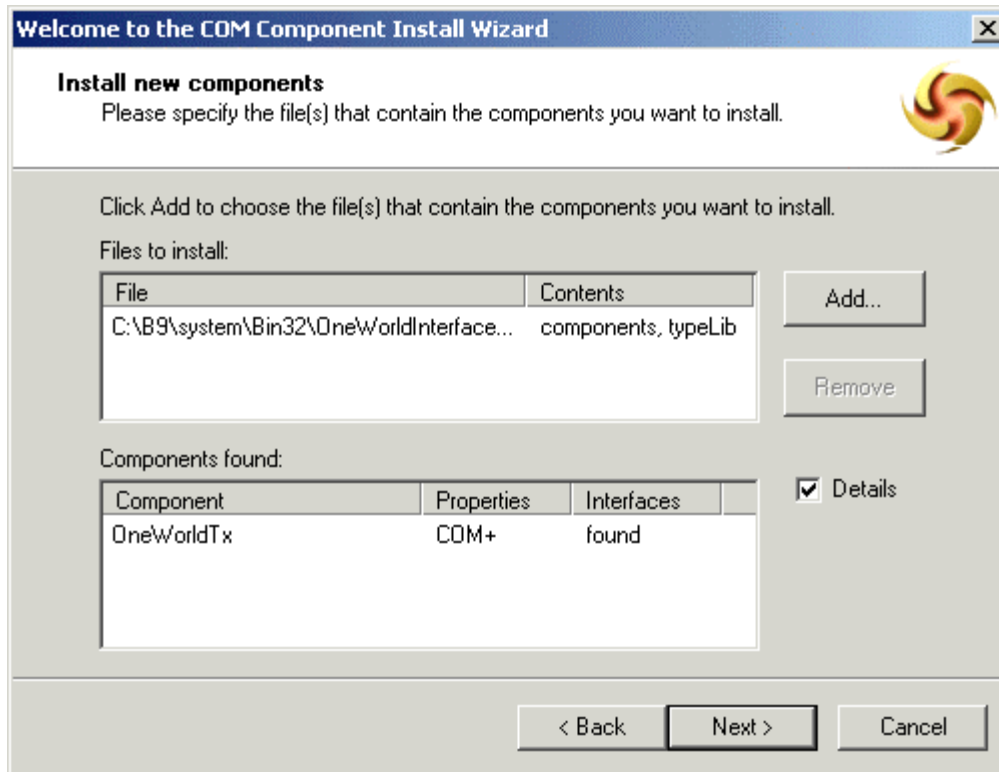
Use the following steps to register OneWorldInterfaceTx.dll:

1. On your PC, navigate to COM+ Applications:
Control Panel > Administrative Tools > Component Services
2. Expand the following icons and folders:
Component Services > Computers > My Computer
3. Choose COM+ Applications.
4. Right-click COM+ Applications, choose New, and then choose Application.
The COM Application Install Wizard appears.

5. On Install or Create a New Application, choose Create an empty application and then click Next.



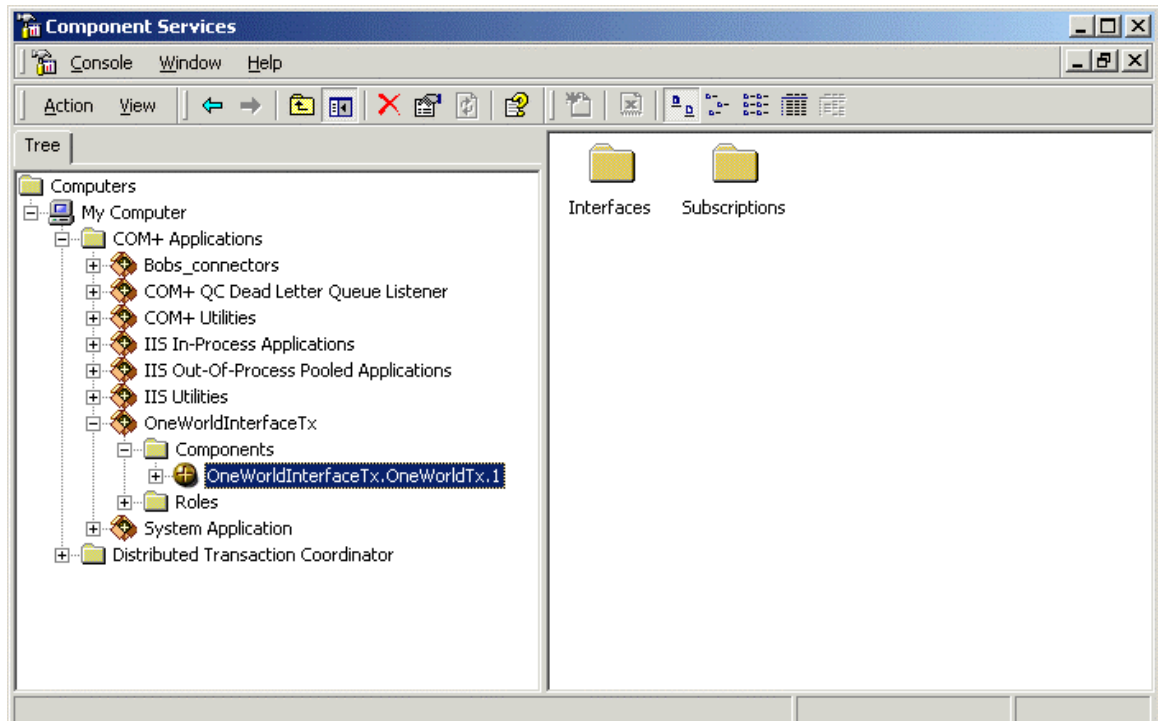
6. On Create Empty Application, enter the name of the application (OneWorldInterfaceTx) that you are registering.
7. Choose an Activation type, and then press Next.
8. On Set Application Identity, choose Interactive User, and then click Next.
9. Click Finish to close the wizard.
10. On your PC, expand the following folders:
COM+ Applications > OneWorldInterfaceTx
11. Choose Components.
12. Right-click Components, choose New, and then choose Component.
The COM Component Install Wizard appears.
13. On Import or Install a Component, choose Install New Component(s), and then click Next.
14. On Select New Files to Install, browse to the application (OneWorldInterfaceTx.dll) on the client install directory or the COM interoperability server.



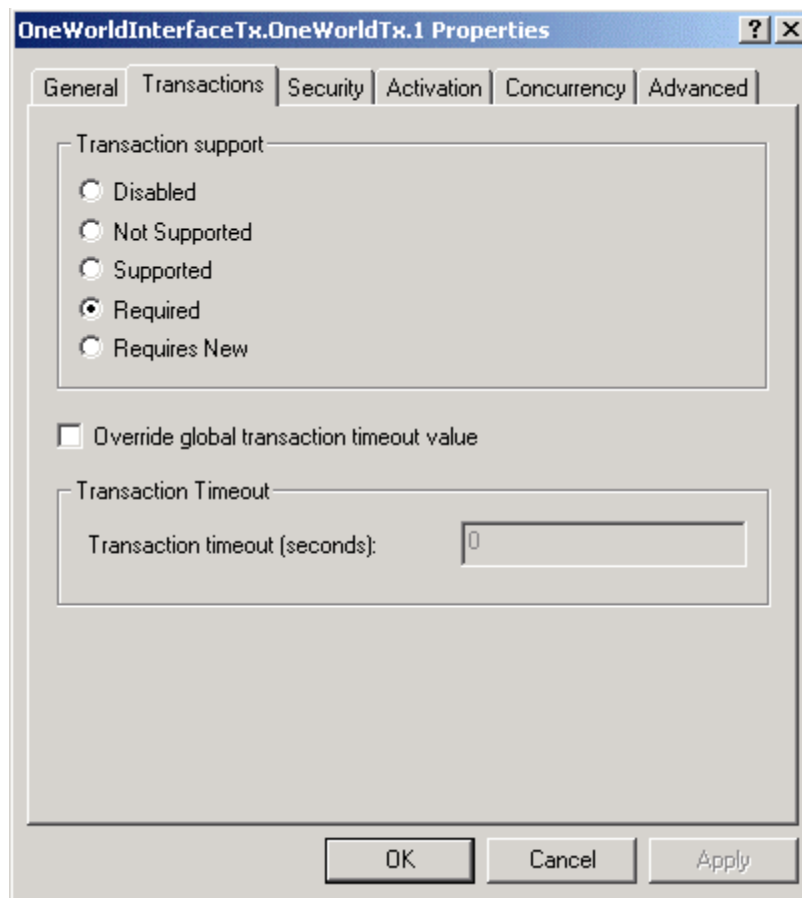
15. Add the application and then click Next.

16. Click Finish to close the wizard.

Your application (OneWorldInterfaceTx.dll) is registered.



17. On your PC, expand the Components folder and then right-click the application (OneWorldInterfaceTx.dll) you just registered.
18. Choose Properties.



19. On OneWorldInterfaceTx Properties, click the Transactions tab.
20. For the Transaction support field, choose the Required option.
21. Click OK.
22. Close the component servers.

The COM connector should be registered using the method described in the chapter titled Installing COM Connector on a Non-PeopleSoft EnterpriseOne Client Environment.

The SalesOrderEntry and other wrapper dlls should be registered using the standard RegSvr32 command.

A new transactional object that is going to participate in the COM+ transactions (for example, SOEClass2.dll) must be created and registered through the COM+ component services of the administrative tools (follow the steps above). The transactions property of this object should be set to Required. This transactional object will use the new OneWorldInterfaceTx.dll for starting a transaction, executing a business function, and so on. The code outline is explained in Case1: PeopleSoft EnterpriseOne Participates in COM+ Transaction. Detail sample code for the SalesOrderEntry transaction object (SOETxObject) is provided.

After the transactional object is created, open a new VB sample SalesOrderEntry client and call the SOEClass2 object. The VB SOETxClient code is provided.

Two cases of the Sales Order Entry application are discussed. Case 1 is when PeopleSoft EnterpriseOne participates in the COM+ transaction. Case 2 is when PeopleSoft EnterpriseOne participates in a distributed transaction.

Case 1: PeopleSoft EnterpriseOne Participates in COM+ Transaction

The following code outline explains how to develop code for COM connector and PeopleSoft EnterpriseOne participation in COM+ transactions:

Code	Explanation
Dim ow As OneWorldTx	Declare new OneWorldTx.
Set ow = New OneWorldTx ow.Initialize laccessNumber, connRole	Initialize the transaction by passing the access number returned from a successful logon and the connector.
ow.BeginTransaction laccessNumber, connRole, 1	Start a transaction in Manual Commit. 1 Manual commit 0 Auto Commit
EditLine, EndDoc	Do all the processing here like BeginDoc.
GetObjectContext().SetComplete Or GetObjectContext().SetAbort	Use SetComplete to commit the transaction through DTC or use SetAbort to abort the transaction.

Note

In COM+, an AutoCommit attribute exists that implicitly commits a transaction if no errors exist. This attribute is in the Component Services Administration tool. However, if an explicit call to SetAbort is made, the transaction aborts.

The following code examples show you how to create a sales order entry transactional object (SOETxObject) and a sales order entry transactional client (SOETxClient). After you create the transactional object and transactional client, you can run the transactions. Use the following steps to run a sales order entry transaction in COM+ where the COM connector and PeopleSoft EnterpriseOne participate:

1. Run the SOETxObject.
2. Run the SOETxClient.
3. Note the Sales Order Entry number that is displayed.
4. When the message box appears for Commit or Abort, choose the appropriate action.
5. Verify in EnterpriseOne whether the sales order has been entered. The sales order should be entered only when committed.

Example Code: Sales Order Entry Transactional Object (SOETxObject)

The following sample code shows how to create a SalesOrderEntry transactional object.

```
Public Sub run()
On Error GoTo errorHandler
Dim ow As OneWorldTx

    Dim bhvr As IOneWorldBHVRCOM

    Dim conn As New Connector          '// COM Connector
    Dim connRole As IConnector2       '// Connector Interface with Roles

    Dim soeObject As JDESalesOrderEntry '// SalesOrderEntry
    Dim soeBeginDoc As D4200310H
    Dim soeEndDoc As D4200310G
    Dim soeEditLine As D4200310F
    Dim soeClearWF As D4200310I
    Dim s As String
    Dim d As New MathNumeric
    Dim mnQuantityOrdered As New MathNumeric
    Dim mnUnitPrice As New MathNumeric
    Dim response

    Dim laccessNumber As Long

    ' Name Information
    Dim strComputerName As String
    Dim lngNameLength As Long

    Const WRITE_FLAG = "2"

    Dim i As Boolean
    Set connRole = conn
    laccessNumber = connRole.Login("UserID", "PWD", "ENV", "*ALL")

    Set ow = New OneWorldTx

    ow.Initialize laccessNumber, connRole
    'oneworld transaction initialized to manual
    ow.BeginTransaction laccessNumber, connRole, 1

    Set bhvr = ow
    bhvr.szApplication = "COM+"
    Set soeObject =
connRole.CreateBusinessObject("SalesOrderEntry.JDESalesOrderEntry", laccessNumber)
' please change the progid to correct progId
    Set soeBeginDoc = soeObject.CreateF4211FSBeginDocParameterset
    Set soeEditLine = soeObject.CreateF4211FSEditLineParameterset
    Set soeEndDoc = soeObject.CreateF4211FSEndDocParameterset
    Set soeClearWF = soeObject.CreateF4211ClearWorkFileParameterset

    ' Get computer name for use later
    strComputerName = Space(30)
```

```

lngNameLength = 30
p_ret = GetComputerName(strComputerName, lngNameLength)
If p_ret <> 1 Then
    MsgBox ("GetComputerName failed!")
    'End
Else
    strComputerName = Mid(strComputerName, 1, lngNameLength)
End If
' MsgBox ("Create Biz Object Done!")

'//////////BEGIN DOC//////////
soeBeginDoc.Reset
soeBeginDoc.cCMDocAction = "A"
soeBeginDoc.cCMPProcessEdits = "1"
soeBeginDoc.cCMUpdateWriteToWF = WRITE_FLAG
soeBeginDoc.szCMPProgramID = "VB"
soeBeginDoc.szCMVersion = "ZJDE0001"
soeBeginDoc.szOrderCo = "00200"
soeBeginDoc.szOrderType = "SO"
szBUnit = "M30"
soeBeginDoc.szBusinessUnit = Space(12 - Len(szBUnit)) + szBUnit
d = Val("4242")
soeBeginDoc.mnAddressNumber = d
soeBeginDoc.mnShipToNo = d
soeBeginDoc.jdOrderDate = Date
soeBeginDoc.cMode = "F"
soeBeginDoc.szUserID = "JDE"
soeBeginDoc.cRetrieveOrderNo = "1"

If strComputerName <> "" Then
    soeBeginDoc.szCMComputerID = strComputerName
End If
' MsgBox ("Before F4211FSBeginDoc")
soeObject.F4211FSBeginDoc soeBeginDoc, ow, connRole, laccessNumber

MsgBox Round(soeBeginDoc.mnOrderNo, 0)

'//////////EDIT LINE//////////

soeEditLine.mnCMJobNo = soeBeginDoc.mnCMJobNumber
orderNum = soeBeginDoc.mnOrderNo
soeEditLine.mnOrderNo = soeBeginDoc.mnOrderNo
soeEditLine.szBusinessUnit = soeBeginDoc.szBusinessUnit
soeEditLine.szCMComputerID = soeBeginDoc.szCMComputerID
soeEditLine.cCMWriteToWFFlag = WRITE_FLAG

soeEditLine.szOrderType = soeBeginDoc.szOrderType
' Load items from UI into edit line structure
soeEditLine.szItemNo = "1001"
mnQuantityOrdered = "2"
soeEditLine.mnQtyOrdered = mnQuantityOrdered

' MsgBox ("Before F4211FSEditLine.")
' Call business function
soeObject.F4211FSEditLine soeEditLine, ow, connRole, laccessNumber

```

```

' MsgBox ("After F4211FSEditLine.")

'//////////ENDDOC//////////
soeEndDoc.mnCMJobNo = soeBeginDoc.mnCMJobNumber
soeEndDoc.mnSalesOrderNo = soeBeginDoc.mnOrderNo
soeEndDoc.szOrderType = soeBeginDoc.szOrderType
soeEndDoc.szCMComputerID = strComputerName
soeEndDoc.cCMUseWorkFiles = WRITE_FLAG
'Call business function

'MsgBox ("Before F4211FSEndDoc.")
soeObject.F4211FSEndDoc soeEndDoc, ow, connRole, laccessNumber
'MsgBox ("After F4211FSEndDoc.")
MsgBoxRes = MsgBox("Do you want to abort?", vbYesNo, "Transaction Decision")
If MsgBoxRes = vbYes Then
    GetObjectContext.SetAbort
Else
    GetObjectContext.SetComplete
    MsgBox ("Order Saved")
End If

'//////////CLEAR WORK FILE//////////

soeClearWF.cClearDetailWF = WRITE_FLAG
soeClearWF.cClearHeaderWF = WRITE_FLAG
soeClearWF.mnJobNo = soeBeginDoc.mnCMJobNumber
soeClearWF.szComputerID = strComputerName
'Call business function
'MsgBox ("Before F4211ClearWorkFile.")
ow.BeginTransaction laccessNumber, connRole, 0
soeObject.F4211ClearWorkFile soeClearWF, ow, connRole, laccessNumber
'MsgBox ("After F4211ClearWorkFile.")
Set soeObject = Nothing
Set soeBeginDoc = Nothing
Set soeEditLine = Nothing
Set soeEndDoc = Nothing
Set ow = Nothing
connRole.Logoff (laccessNumber)
Set connRole = Nothing

Exit Sub

errorhandler:
    GetObjectContext().SetAbort
    connRole.Logoff (laccessNumber)
    Set ow = Nothing
End Sub

```

Example Code: Sales Order Entry Transactional Client (SOETxClient)

The following sample code shows how to create a SalesOrderEntry transactional client.

```
'\\\\\\\\SOETxClient\\\\\\\\
Private Sub Form_Load()
Dim c As SOEClass2                '// VB SOE transactional object
Set c = New SOEClass2
c.run
Set c = Nothing
End Sub
```

Case 2: PeopleSoft EnterpriseOne Participates in a Distributed Transaction

Case 2 explains how a Sales Order Entry and test SQL database query is done as a distributed transaction using COM+.

The following sample code, called MTStest.vbp, shows how to create a distributed transaction using COM+. This project contains the following two classes:

- MTTestClass, which queries and updates a test SQL database
- OWTxClass, which runs the Sales Order Entry

OWTxClass is almost identical to the previous SOETxObject, except that the message box for commit or abort is no longer necessary.

MTStest.dll must be registered in the COM+ Component Services, and the transaction property should be set to required; it might have been set already.

Create a sample SQL test database table SOE2PCTest. SOE2PCTest table will have two columns, SONum and LastSONum. The test chooses the LastSONum and then updates the table by incrementing the previous value by 1 when commit is called.

Sample code called ClientPrj.vbp will call the transactional object.

Both of the transactions are committed by the DTC when the SetComplete call is made. The DTC aborts the transaction when the SetAbort call is made or if any part of the transaction fails.

Use the following steps to run a sales order entry as a distributed transaction in COM+ where the COM connector, PeopleSoft EnterpriseOne, and an SQL database participate.

1. Run the MTStest.vbp.
2. Run the ClientPrj.vbp.
3. Click the Call Database_Test_Method button.
4. Switch back to the MTStest and note the sales order number.
5. When a message box appears to Commit or Abort, choose the appropriate action.
6. Verify in PeopleSoft EnterpriseOne whether the sales order has been entered. When the transaction is aborted, the sales order should not be in PeopleSoft EnterpriseOne, and the test database should not increment the count.

Example Code: MTStest for Distributed Transaction

The following code sample provides detail code for creating MTStest.

Note

The following sample code has message box statements to help better understand the step-by-step flow of the code. Since DTC is managing the transactions, it is necessary not to lock the tables for a long time. When you use message boxes, you stop the program flow. When regression testing, you must remove all of the message boxes. You can write to a log file instead.

```
Option Explicit

Public Function Database_Test_Method( _
ByVal szConnect As String) As String

Dim stmt As String

On Error GoTo errhandler

Dim ctxObject AsObjectContext
Set ctxObject = GetObjectContext()

Dim MsgBoxRes
Dim cn As ADODB.Connection
Dim rsSelect As ADODB.Recordset
Dim rs As ADODB.Recordset

Set cn = New ADODB.Connection
With cn
    .ConnectionTimeout = 10
    .ConnectionString = szConnect
    .Open
End With

' .....
' SONUM and LASTSONUM are columns created in a database called COMPLUS. '
' Database server is called soe2ptest. '
' LASTSONUM gets incremented when commit is used. '
' Change these values according to Database created '
' .....

Set rs = New ADODB.Recordset
Set rsSelect = New ADODB.Recordset
rsSelect.Open "SELECT LASTSONUM FROM soe2ptest", cn, adOpenDynamic, _
adLockReadOnly
Dim i As Integer
For i = 1 To 3

    stmt = "Update SOE2PCTest set LASTSONUM=" & rsSelect(0).Value + 1 & " where
SONUM = 1"
    cn.Execute stmt, 1, -1
    rsSelect.Close
```

```

Dim c As OWTXClass
Set c = New OWTXClass

c.run

Set c = Nothing
cn.Close

Set rs = Nothing
Set cn = Nothing
MsgBoxRes = MsgBox("Do you want to Commit?", vbYesNo, "Transaction Decision")
If MsgBoxRes = vbYes Then
    ctxObject.SetComplete
Else
    ctxObject.SetAbort
End If
Next I

Exit Function

errhandler:
    Err.Raise vbObjectError, "MTSTest.MTStest.Database_Test_Method", _
    Err.Description
    ctxObject.SetAbort
    Exit Function

End Function

```

Example Code: ClientPrj for Distributed Transaction

The following code sample provides detail code for creating ClientPrj.

Note

The following sample code has message box statements to help better understand the step-by-step flow of the code. Since DTC is managing the transactions, it is necessary not to lock the tables for a long time. When you use message boxes, you stop the program flow. When regression testing, you must remove all of the message boxes. You can write to a log file instead.

```

Private Sub Command2_Click()
    Dim szConnect As String
    szConnect = "Driver={SQL Server};" & _
        "Server=AServerName;Uid=UserID;Pwd=Passwd;Database=DBName"
    '(NOTE: You may need to change the connection
    ' information to connect to your database.)

    Dim obj As Object
    Set obj = CreateObject("MTStest.MTSTestClass")

    MsgBox obj.Database_Test_Method(szConnect)
    Set obj = Nothing
    Unload Me
End Sub

Private Sub Form_Load()

    Command2.Caption = "Call Database_Test_Method"

End Sub

```

OCM Support for the COM Connector

You use Object Configuration Manager (OCM) to map business functions to a PeopleSoft EnterpriseOne server so that the COM connector can access OCM to run business functions. You no longer configure the jdeinterop.ini file to define the PeopleSoft EnterpriseOne server from which you want to execute business functions. Using OCM support should result in increased performance, scalability, and load balancing. OCM mapping enables the COM interoperability server to distribute the processes of the COM connector client to various PeopleSoft EnterpriseOne servers' requests, depending on the user, environment, and role name.

To take the advantage of COM connector OCM support, your system administrator should do the following:

- Get the GenCOM PeopleSoft EnterpriseOne 8.9 (or later) version and regenerate the business wrapper function.
- Configure your OCM and map your business function on the enterprise server.
- Add the following settings in your jdeinterop.ini configuration file.

[INTEROP]

Setting	Explanation
EnterpriseServer = ntropt1	For COM events and backward compatibility.
SecurityServer = ntropt1	Validates the login.
Port = 6079	The port number.

Your database administrator or PeopleSoft EnterpriseOne administrator can provide the following settings for the [OCM] section of the jdeinterop.ini configuration file. This information is used for database connectivity.

[OCM]

Setting	Explanation
DSN=ODA ITTND17	The data source name from the system DSN of your ODBC setting.
OCM Datasource = COM OCM	System data source for PeopleSoft EnterpriseOne client.
DB User = JDE	User for the data source connection.
DB Pwd = JDE	Password for the data source connection.
Object Owner = SYS9	For UNIX platforms, this is the object owner in the [DB SYSTEM SETTINGS].
Seperator=.	For Oracle, SQL and UDB databases, the separator is a period (.); for iSeries, the separator is a "/".

If you use a client machine, the settings can be found in your client jde.ini file. The following examples show the database name and object owner:

- JDE9.SYS9

Where JDE9 is the database name and SYS9 is the object owner.

See Also

The following topics in the *Configurable Network Computing Implementation Guide*:

- ❑ *Object Configuration Manager* for more information about OCM
- ❑ *Mapping Objects* for information about mapping the OCM

Installation Information

Because the GenCOM application produces interfaces based on the package currently installed on the machine, installation plans must be made on a site-by-site basis. The DLLs produced are business function release-dependent and can only be installed on machines with the identical packages available.

The GenCOM output is COM servers in the form of DLLs. You can use these to create an interface with the PeopleSoft EnterpriseOne system. You should not assume that a client has installed these servers as part of the standard PeopleSoft EnterpriseOne installation. You should provide a full installation of any of the servers your applications require.

Using COM Tracing and Logging

You can use COM tracing and logging to help you debug your COM applications. Tracing and logging are configurable through the jdeinterop.ini file. The logging format is similar to the PeopleSoft EnterpriseOne logging format. For example, both logging formats include the Time Thread ID [User ID] and Description:

Thu Mar 02 14:48:01 2000 294 [AR618238] Failed to Login to Environment <ADEVHPO2>

Errors are written to the JobFile and trace messages are written to the DebugFile. When trace is enabled, error messages go into both trace and error logs.

You can change the jdeinterop.ini settings while the connector is running by completing the following the steps:

1. Modify the jdeinterop.ini file.
2. Right-click the Connector System Tray Icon.
3. Choose the menu item ChangeIniSettings.

If an option in the jdeinterop.ini file does not have an entry, the default value is used.

Troubleshooting

Tracing affects performance. You do not need tracing turned on unless you are debugging. If your performance is negatively affected, ensure that you do not have your tracing level set to a nonzero number.

If no logs are generated, complete the following steps:

- Ensure that you have specified the proper path in the ini file.
- Verify that disk space and the permissions on your file system are correct.
- Verify whether the default log files have been generated.
- Check the interop.log to see if any errors corresponding to logging have been generated.
- Check the interop.log file to see if the ini settings that are being used are the same as what you have specified elsewhere.

COM Reliability

Graceful fail-over and fault tolerance mechanisms are important, especially for applications that require high availability. The COM connector provides basic support for fault tolerance at the protocol level.

You should take additional precautions to provide further reliability. After you use the COM connector to enter an order or execute a business function, your process should:

- Handle transaction failures. Transactions can fail because of communication line failures. Sometimes transactions must be aborted because of errors in input or deadlocks. These failures must be handled appropriately.
- Wait for the confirmation or success notification from the business function to ascertain that the call was successfully committed.
- Query on the order entered to make sure that it has been committed to the database. Due to high network traffic, a business function can properly execute, but the confirmation message might not reach the user.

COM Connector Events

The COM connector events solution uses Microsoft's COM+ Events Service. COM+ Events Loosely Coupled Events, which matches and connects publishers and subscribers, is part of the Windows2000 Component Services. The EventClass is a COM+ component that contains interfaces and methods that are used by the publisher to fire events. The EventClass manages the connection between publisher and subscribers. PeopleSoft provides the EventClass.dll, which contains the IOWEvent interface. The COM servers and COM clients must implement this interface so that when an event gets fired, this interface is called by the COM+ events and the implementation is executed. The implementation decides what the delivered event and the event data should do. This implementation is COM Server or COM client specific.

Note

You should have a basic understanding of the COM+ Events Service.

COM+ events supports Z events, real-time events, and XAPI events. COM+ events is not dependent on PeopleSoft EnterpriseOne setup for event generation.

See Also

- ❑ *Microsoft COM+ web site* (<http://www.msdn.microsoft.com/>)
- ❑ *Events in the Interoperability Guide* for more information about generating Z, real-time, and XAPI events

Registering Components

So that subscribers can find an event class and subscribe to it, the PeopleSoft EnterpriseOne event class must be registered with COM+. In addition, COM+ requires a type library that describes the event interface and methods so that subscribers and publishers can be properly matched and connected. The type library must reside in or be accompanied by a self-registering DLL.

To register the PeopleSoft EnterpriseOne Events Class with COM+ Services, you must do the following:

- Add a new COM+ application for the PeopleSoft EnterpriseOne event class
- Install the PeopleSoft EnterpriseOne event class

Note

Before you register the PeopleSoft EnterpriseOne Event Class with COM+ Services, set up your COM server. Your COM server can be set up on either a PeopleSoft EnterpriseOne machine or a non-PeopleSoft EnterpriseOne machine, or both.

See Also

- ❑ *Installation Information in the Connectors Guide* for information about setting up your COM connector

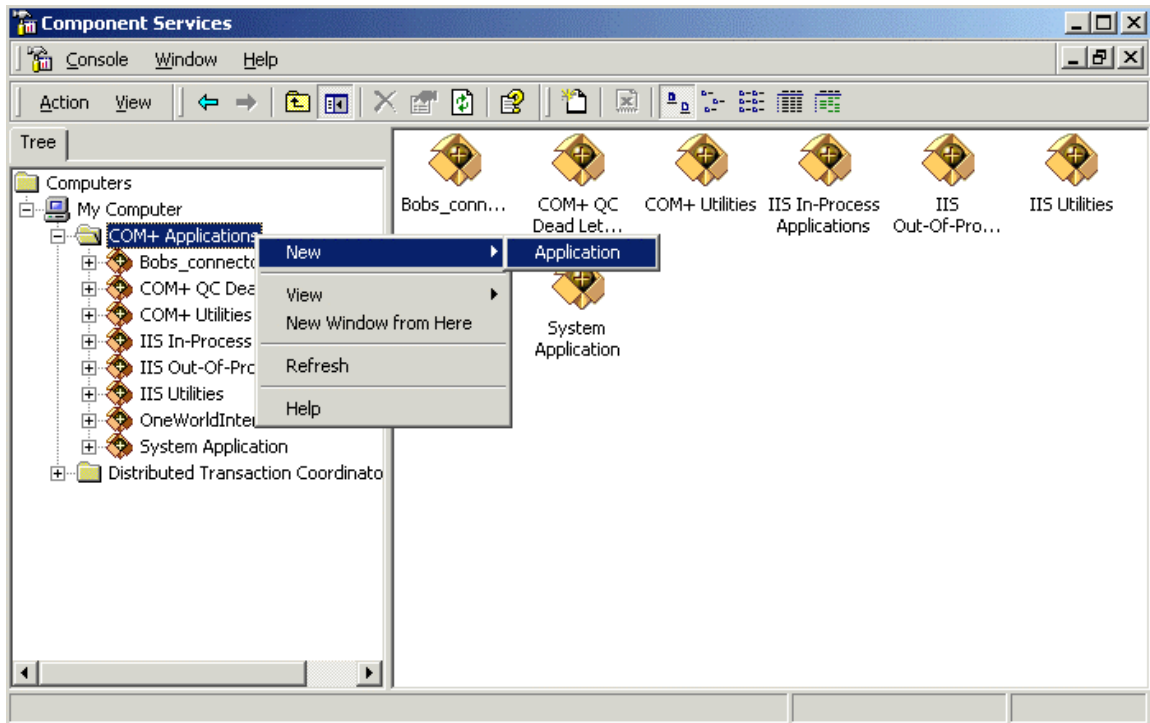
► **To add a new application**

From your Windows 2000 machine, navigate to COM+ Applications:

Control Panel > Administrative Tools > Component Services

Expand the following icons and folders:

Component Services > Computers > My Computer > COM+ Applications



1. Choose COM+ Applications.
2. Right-click COM+ Applications, choose New, and then choose Application.

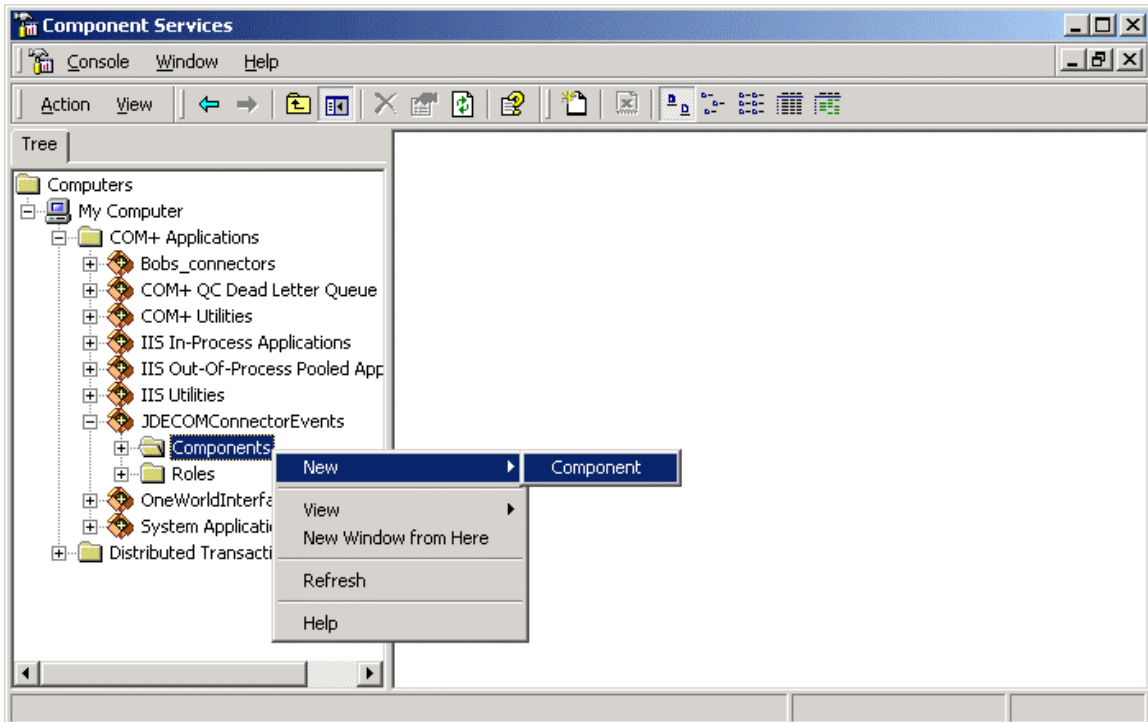
The COM Application Install Wizard appears. The following steps apply to the wizard.

3. On Install or Create a New Application, choose Create an empty application.
4. On Create Empty Application, enter the name of your application (for example, JDECOMConnectorEvents).
5. Choose an option for Activation Type, and then click Next.
6. On Set Application Identity, choose the following option, and then click Next.
 - Interactive User
7. Click Finish.

A new application, with the name you entered in Step 4 above, is added to COM+ Applications.

► **To install the event class**

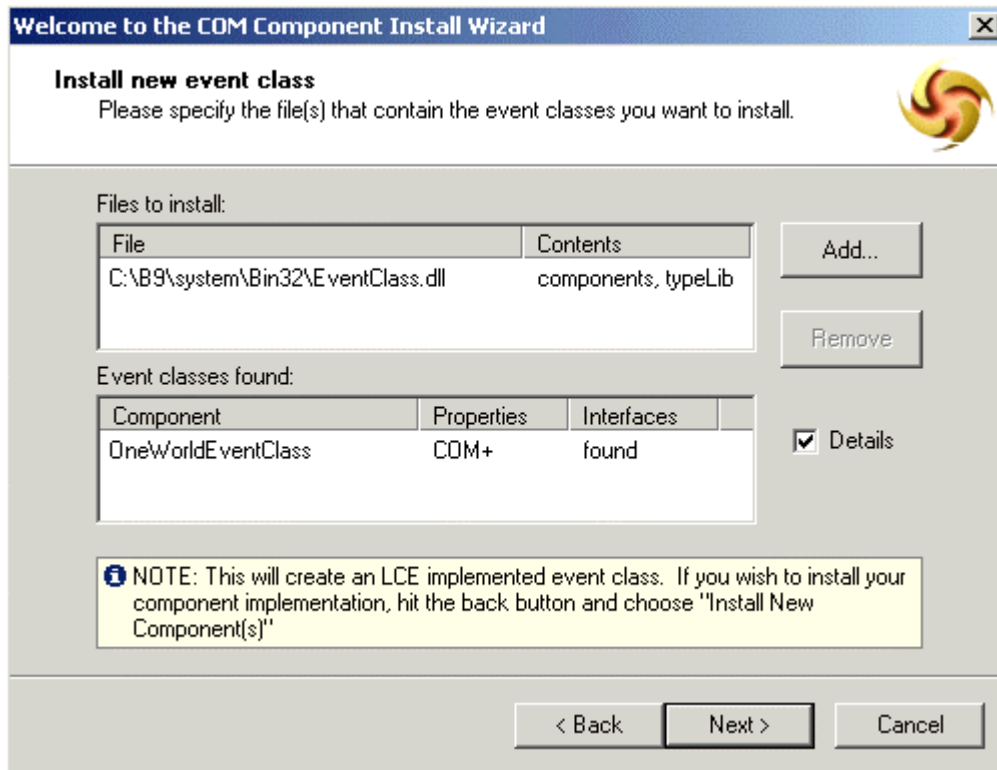
On Component Services, expand the folder for your new application (for example, JDECOMConnectorEvents).



1. Choose Components.
2. Right-click Components, choose New, and then choose Component.

The COM Component Install Wizard appears. The following steps apply to the wizard.

3. On Import or Install a Component, choose Install new event class(es).
4. On Select Files to Install, browse to the EventClass.dll on your Windows 2000 machine.



5. Choose EventClass.dll, and then click Open.

Install new event class appears with information in the following fields:

- Files to install
- Event classes found

6. Click Next, and then click Finish.

EventClass.dll is successfully added to Component Services.

Event Subscriptions

The COM connector supports both persistent and transient event subscriptions from the PeopleSoft EnterpriseOne server. The events are subscribed from the PeopleSoft EnterpriseOne server that is specified in the [INTEROP] section of the jdeinterop.ini file. The events are received through the EventListener. The EventListener runs as long as the COM connector is up and running. The COM connector runs as a small globe icon in the bottom right corner of the Windows taskbar.

You must also set up the [EVENTS] section of the jdeinterop.ini file.

Note

The COM connector does not support subscription of events from multiple PeopleSoft EnterpriseOne servers.

See Also

- ❑ *jdeinterop.ini* in the *Connectors Guide* for information about setting up the [INTEROP] and [EVENTS] sections

Logging COM Events

Logging for COM events is entered in the *interopDebug.log* file. The error log is the *interop.log*.

Implementing the PeopleSoft EnterpriseOne Interface

You must develop an object that implements the *IOWEvent* interface. For further discussion and to provide code samples, the name *EventSink* is used as the object name. The object that you develop to implement the *IOWEvent* can have a different name. *EventSink* implements the *IOWEvent* interface and the method within the interface, and then consumes the PeopleSoft EnterpriseOne event. The *EventSink* implementation is client specific. *EventSink* receives the event from PeopleSoft EnterpriseOne by implementing the interface specified in *EventClass*. The following outline shows how to develop an *EventSink* component:

```
Option Explicit
Implements IOWEvent
Public Event OneWorldEvent(ByVal EventName As String, ByVal Data As String)

Public Sub IOWEvent_OneWorldEvent(ByVal EventName As String, ByVal Data As String)
'// Add code specific to the client implementation here
    RaiseEvent OneWorldEvent(EventName, Data)
End Sub
```

The following list outlines the steps for you to follow to use the *EventManager* library and *MessageHandler* Interface to subscribe to events.

1. Log on to the connector. Successful logon returns an access number.
2. Create the *EventSink* object.
3. Create the *MessageHandler* object.
4. Call methods on the *MessageHandle* for *Subscribe*, *Unsubscribe*, *GetTemplate*, and *GetEventList* for the respective event.
5. To keep the session alive and not timeout from receiving events, call the *UpdateOutBoundSessionTime* method on the connector interface. This method updates the user session time to the current time.
6. To subscribe to the events as persistent, register VB *EventSink* in the COM+ Component Services and add the subscription for the *EventClass*.

Example: Sample Code for Creating EventSink

The following sample code is for creating the EventSink.dll. EventSink implements the EventClass interface, IOEvent(). You can use a name other than EventSink.

EventSink: OneWorldTransientEventSink.cls

```
Option Explicit

Implements IOEvent
Public Event OneWorldEvent(ByVal EventName As String, ByVal Data As String)

Public Sub IOEvent_OneWorldEvent(ByVal EventName As String, ByVal Data As String)
    Dim flsObject As New Scripting.FileSystemObject
    Dim varEventFile As TextStream
    Dim strEventFile As String
    strEventFile = "C:\temp\eventDataPer.xml" ' change this to a valid directory
    If Dir(strEventFile) = "" Then
        Set varEventFile = flsObject.CreateTextFile(strEventFile, False, False)
    Else
        Set varEventFile = flsObject.OpenTextFile(strEventFile, ForWriting, False)
    End If

    varEventFile.WriteLine Data
    varEventFile.Close
    RaiseEvent OneWorldEvent(EventName, Data)
End Sub
```

Registering EventSink for Persistent Subscription

After you register an event class in the COM+ catalog, you can add subscribers to the event class and subscriptions to the subscribers. For persistent event subscription, do the following:

- Add a new application for EventSink.
- Install the type library component for EventSink.
- Add a subscription.

Note

To add EventSink, follow the steps in the task *To add a new application* in the *Connectors Guide*. The name of the application is EventSink, or a name that you prefer.

► To install the EventSink component

On Component Services, expand the folder for your new application (for example, EventSink).

1. Choose Components.
2. Right-click Components, choose New, and then choose Component.

The COM Component Install Wizard appears. The following steps are for the wizard.

3. On Import or Install a Component, choose Install new component(s).
4. On Select Files to Install, browse to the EventSink.dll that you previously developed.
5. Choose EventSink.dll, and then click Open.

Install new component appears with information in the following fields:

- Files to install
- Event classes found

6. Click Next, and then click Finish.

EventSink.dll is successfully added to Component Services.

► To add a subscription

In COM+ Applications, expand the following folders:

JDECOMConnectorEvents > Components > EventSink.OneWorldTransientEventSink

1. Choose Subscription.
2. Right-click Subscription, choose New, and then choose Subscription.

The COM New Subscription Wizard appears. The following steps apply to the wizard.

3. On Select Subscription Method(s), chose IOWEvent, and then click Next.
4. If appropriate, choose the following option:

- Use all interfaces for this component

5. On Select Event Class, choose the event class (for example, PeopleSoft.EventClass.OneWorldEventClass.1), and then press Next.

If multiple EventSink classes have implemented the event interface, then use all event classes that implement that specified interface. If only one EventSink class has implemented the event interface, then just choose that specific class.

6. On Subscription Options, enter the name of the subscription (for example, MySubscription).
7. In the Options area, choose the following option, and then click Next:
 - Enable this subscription immediately
8. Click Finish.

A new subscription, with the name you entered in Step 6, is added to COM+ Services. You must define the name of the event for the subscription.

9. Right-click the subscription (for example, MySubscription), and then choose Properties.
10. On MySubscription Properties, click the Options tab.
11. Chose the following option:
 - Enabled

12. In the Filter criteria field, enter the name of the event for which you want a subscription.

Enter all of the events for which you want to subscribe. The filter criteria string supports relational operations (=, ==, !=, ~, ~=, <>), nested parentheses, and logical words (AND, OR, and NOT); for example:

EventName=='RTSOOUT' OR EventName=="RTPOOUT"

Note

For more information about filtering, see *Subscribe parameter filtering* in the Microsoft COM+ Events online documentation.

13. Click OK.

Example: Sample Code for COMConnector.dll

The following sample code logs on to the COM connector, creates the MessageHandler object, and performs Subscribe, Unsubscribe, GetTemplate, and GetList. Before executing the subscriber, use the Regsvr32 command to register COMConnector.dll.

COMConnector: frmLogin.frm

```
Option Explicit

Public bLoginEnv As Boolean

Private Sub cmdCancel_Click()
    'set the global var to false
    'to denote a failed login
    bLoginEnv = False
    Me.Hide
End Sub

Private Sub cmdOK_Click()
    'check for correct password
    If txtUserName = "" Or txtenvironment = "" Then
        bLoginEnv = False
        MsgBox "Must Enter User Name and Environment to continue"
    Else
        bLoginEnv = True
        Me.Hide
    End If
End Sub
```

COMConnector Common.bas

```
Option Explicit
Dim conn As New Connector
Dim connRole As IConnector2
Dim messageHandler As New messageHandler
Dim mHandlerInterface As IMessageHandler
Dim lngAccessNumber As Long
Public Sub comm_Initialize()
    Set connRole = conn
    frmLogin.bLoginEnv = False
    frmLogin.Show
    While Not frmLogin.bLoginEnv
        DoEvents
    Wend
    lngAccessNumber = connRole.Login(frmLogin.txtUserName, frmLogin.txtPassword,
    frmLogin.txtenvironment, frmLogin.txtrole)
    Set mHandlerInterface = messageHandler
End Sub

' NOTE: the code in this module is particular to this prototype.  Different code
' would be used in a production version to send messages to PeopleSoft
' EnterpriseOne via PeopleSoft communication protocols

Public Sub SendSubscriptionToWorld(eventName As String, oneworldevent As
IOEvent, mode As Long)
    mHandlerInterface.SubscribeEvent lngAccessNumber, conn, eventName,
oneworldevent, mode
End Sub
Public Sub SendUnSubscribeToWorld(eventName As String, oneworldevent As
IOEvent, mode As Long)
    mHandlerInterface.UnSubscribeEvent lngAccessNumber, conn, eventName,
oneworldevent, mode
End Sub
Public Sub getEventListFromOneWorld(eventList As String)
    mHandlerInterface.GetEventList lngAccessNumber, conn, eventList
End Sub
Public Sub getEventTemplateFromOneWorld(eventName As String, eventTemplate As
String)
    mHandlerInterface.GetEventTemplate lngAccessNumber, eventName, conn,
eventTemplate
End Sub
```

COMConnector : SubscriptionManager

```
Option Explicit

Private m_SubscribedEvents As Collection

Private Sub Class_Initialize()
    Set m_SubscribedEvents = New Collection
    comm_Initialize
End Sub

Public Sub GetEventList(eventList As String)
    getEventListFromOneWorld eventList
End Sub

Public Sub CreateTransientSubscription(eventName As String, oneworldevent As IOEvent)
    SubscribeToOneWorldEvent eventName, oneworldevent, 0
End Sub

Public Sub CreatePersistentSubscription(eventName As String, oneworldevent As IOEvent)
    SubscribeToOneWorldEvent eventName, oneworldevent, 1
End Sub

Public Sub RemoveTransientSubscription(eventName As String, oneworldevent As IOEvent)
    UnSubscribeToOneWorldEvent eventName, oneworldevent, 0
End Sub

Public Sub RemovePersistentSubscription(eventName As String, oneworldevent As IOEvent)
    UnSubscribeToOneWorldEvent eventName, oneworldevent, 1
End Sub

Public Sub GetEventTemplate(eventName As String, eventTemplate As String)
    getEventTemplateFromOneWorld eventName, eventTemplate
End Sub

Public Sub SubscribeToOneWorldEvent(eventName As String, oneworldevent As IOEvent, mode As Long)
'Private Function SubscribeToOneWorldEvent(EventName As String) As Boolean
    ' we've already subscribed if the subscription is in our list
    Dim alreadySubscribed As Boolean
    alreadySubscribed = (CollectionContainsString(m_SubscribedEvents, eventName) = True)

    ' now do the right thing...
    If (alreadySubscribed = False) Then
        ' this instance of the COMConnector has not seen this event
        ' before, so add it to our list...
        m_SubscribedEvents.Add (eventName)

        ' ...and go ahead and subscribe to the event from PeopleSoft
        ' EnterpriseOne
        SendSubscriptionToOneWorld eventName, oneworldevent, mode
    End If

    'SubscribeToOneWorldEvent = alreadySubscribed
End Sub
```

```

Private Function CollectionContainsString(col As Collection, str As String)
    Dim colItem As Variant
    For Each colItem In col
        If (colItem = str) Then
            CollectionContainsString = True
            Exit Function
        End If
    Next
    CollectionContainsString = False
End Function

Public Sub UnSubscribeToOneWorldEvent(eventName As String, oneworldevent As
IOWEvent, mode As Long)
Dim alreadySubscribed As Boolean
    alreadySubscribed = (RemoveFromCollection(m_SubscribedEvents, eventName))
    If (alreadySubscribed = False) Then
        MsgBox "Event Not Subscribed"
    Else
        ' ...and go ahead and subscribe to the event from
        ' PeopleSoft EnterpriseOne
        SendUnSubscribeToOneWorld eventName, oneworldevent, mode
    End If
' End If
End Sub

Private Function RemoveFromCollection(col As Collection, str As String)    Dim
colItem As Variant
    Dim count As Integer
    count = 0
    For Each colItem In col
        count = count + 1
        If (colItem = str) Then
            col.Remove count
            RemoveFromCollection = True
            Exit Function
        End If
    Next
    RemoveFromCollection = False
End Function

```

Example: Sample Code for Subscriber

Subscriber is the GUI that gets the EventsList, EventTemplate, Subscribe, and Unsubscribe. Subscriber is built as a VB executable. Typical usage is to get the EventList first, which populates the dropdown list box with the list of events that are supported by the PeopleSoft EnterpriseOne server. Choose the event that needs to be subscribed from the PeopleSoft EnterpriseOne server and the type of subscription. Click Subscribe to add a Subscription, or click Unsubscribe to unsubscribe from the PeopleSoft EnterpriseOne server. The Subscribed events and the Received events are shown in separate boxes. The received event is displayed in the window on the right. The event received can be integrated with BizTalk by choosing the Enable BizTalk Integration option. You should have previously set up BizTalk; if not already installed, install the BizTalk Server 2000 Developer. If the Module 1 tutorial in the BizTalk Server documentation runs properly, then the BizTalk Server is properly installed. Before building the subscriber, you should use the Regsvr32 command to register EventSink.dll and COMConnector.dll.

Subscriber : MainForm.frm

The following code sample is for the GUI and the control buttons on the GUI. The following code should be built along with the BizTalk.cls, after registering the COMConnector.dll and MyEventSink.dll.

```

Option Explicit

' ----- ** -----
'
'                               Member Variables
'
' ----- ** -----

Private m_SubscriptionManager As SubscriptionManager
Private WithEvents m_OneWorldTransientEventSink As OneWorldTransientEventSink
Private Sub Combo1_Change()

End Sub
Private Sub Check1_Click()

End Sub

Private Sub btnClear_Click(Index As Integer)
    lvwReceivedEvents.ListItems.Clear
End Sub

' ----- ** -----
'
'                               GetEventTemplate
'
' ----- ** -----
Private Sub btnGetEventTemplate_Click()
    Dim EventName As String
    Dim EventTemplate As String
    EventName = cEventList.List(cEventList.ListIndex)
    m_SubscriptionManager.GetEventTemplate EventName, EventTemplate
    Dim flsObject As New Scripting.FileSystemObject
    Dim varTemplateFile As TextStream
    Dim strTemplateFile As String
    strTemplateFile = "C:\temp\event_template.xml"
    If Dir(strTemplateFile) = "" Then
        Set varTemplateFile = flsObject.CreateTextFile(strTemplateFile, False,
False)
    Else
        Set varTemplateFile = flsObject.OpenTextFile(strTemplateFile, ForWriting,
False)
    End If

    varTemplateFile.WriteLine EventTemplate
    varTemplateFile.Close

    wbEventData.Navigate "c:\temp\event_template.xml"
End Sub

' ----- ** -----
'
'                               Event Handlers
'
' ----- ** -----

```

```

Private Sub Form_Load()
    Set m_SubscriptionManager = New SubscriptionManager
    Set m_OneWorldTransientEventSink = New OneWorldTransientEventSink

    EnableBizTalkIntegrationGroup
End Sub

Private Sub m_OneWorldTransientEventSink_OneWorldEvent(ByVal EventName As String,
ByVal Data As String)
    ' add the event name and payload to the list
    Dim mTempItem As ListItem
    Set mTempItem = lvwReceivedEvents.ListItems.Add()
    mTempItem.Text = EventName
    'mTempItem.SubItems(1) = Data
    Dim flsObject As New Scripting.FileSystemObject
    Dim varEventFile As TextStream
    Dim strEventFile As String
    strEventFile = "C:\temp\eventData.xml"
    If Dir(strEventFile) = "" Then
        Set varEventFile = flsObject.CreateTextFile(strEventFile, False, False)
    Else
        Set varEventFile = flsObject.OpenTextFile(strEventFile, ForWriting, False)
    End If

    varEventFile.WriteLine Data
    varEventFile.Close
    wbEventData.Navigate "c:\temp\eventdata.xml"

    ' send the event to BizTalk (if it's enabled)
    If (chkEnableBizTalkIntegration.Value = Checked) Then
        Dim oBizTalk As BizTalk
        Set oBizTalk = New BizTalk
        oBizTalk.RunSchedule txtScheduleFile.Text, Data
    End If
End Sub

'----- ** -----
'
'                               GetEventList
'----- ** -----
Private Sub btnGetEventList_Click()
    Dim events As String
    Dim myValue As String
    Dim myString As String
    Set m_SubscriptionManager = New SubscriptionManager
    m_SubscriptionManager.GetEventList events

    cEventList.Clear
    myString = events
    Do Until events = ""
        If InStr(1, myString, ":") > 0 Then
            myValue = Left(myString, InStr(1, myString, ":") - 1)
            myString = Mid(myString, InStr(1, myString, ":") + 1)
        End If
    Loop

```

```

        Else
            myValue = myString
            events = ""
        End If

        cEventList.AddItem myValue
    Loop
    cEventList.ListIndex = 0
End Sub

'----- ** -----
'
'                Subscribe Event
'
'----- ** -----

Private Sub btnSubscribe_Click()
    ' subscribe to the named event.
    Dim eventName As String
    eventName = cEventList.List(cEventList.ListIndex)
    If (chkPersist.Value = Checked) Then
        m_SubscriptionManager.CreatePersistentSubscription eventName,
m_OneWorldTransientEventSink
    Else
        m_SubscriptionManager.CreateTransientSubscription eventName,
m_OneWorldTransientEventSink
    End If
    Dim mTempItem As ListItem
    Set mTempItem = lvwSubscribedEvents.ListItems.Add()
    mTempItem.Text = eventName
End Sub

'----- ** -----
'
'                UnSubscribe Event
'
'----- ** -----

Private Sub btnUnsubscribe_Click()
    Dim eventName As String
    eventName = cEventList.List(cEventList.ListIndex)
    Dim lstItem As ListItem
    Dim count As Integer
    Dim found As Boolean
    count = 0
    found = False
    For Each lstItem In lvwSubscribedEvents.ListItems
        count = count + 1
        If lstItem = eventName Then
            lvwSubscribedEvents.ListItems.remove (count)
            GoTo remove
            found = True
        End If
    Next
    If found = False Then
        MsgBox "Event Not Subscribed"
    End If

```

```

        End If
remove: If (chkPersist.Value = Checked) Then
    m_SubscriptionManager.RemovePersistentSubscription EventName,
m_OneWorldTransientEventSink
    Else
        m_SubscriptionManager.RemoveTransientSubscription EventName,
m_OneWorldTransientEventSink
    End If
End Sub

Private Sub chkEnableBizTalkIntegration_Click()
    EnableBizTalkIntegrationGroup
End Sub
'----- ** -----
'
'                                Clear the Received Events List
'----- ** -----

Private Sub btnClear0_Click()
    ' clear the events from the list
    lvwReceivedEvents.ListItems.Clear
End Sub

Private Sub btnClose_Click()
    Unload Me
End
End Sub

' ----- ** -----
'
'                                Private Functions
' ----- ** -----

Private Sub Initialize()
    ' Create the event sink
    Set m_OneWorldTransientEventSink = New OneWorldTransientEventSink
End Sub

Private Sub EnableBizTalkIntegrationGroup()
    Dim blnEnable As Boolean
    blnEnable = (chkEnableBizTalkIntegration.Value = Checked)
    lblScheduleFile.Enabled = blnEnable
    txtScheduleFile.Enabled = blnEnable
End Sub

```

Example: Sample Code for BizTalk Integration

The following code is for the BizTalk integration for the received event.

Subscriber: BizTalk.cls

```
Option Explicit

'*****
'***** ExecuteTutorial
'*****
'***** Purpose: This component is used to exercise
'*****           the XLANG schedule portion of tutorial accompanying
'*****           BizTalk Server (this is the Module 1 Tutorial).
'*****           The component launches the specified schedule
'*****           file and passes the data file specified
'*****           to it via MSMQ.
'*****
'*****           NOTE: the source code in this component is a direct
'*****           adoption of the code found in the Module 1 Tutorial in the
'*****           BizTalk Server 2000 documentation.
'*****           The default location for the original version of this
'*****           source is found in: C:\Program Files\Microsoft BizTalk
Server\Tutorial
'*****           \Schedule\Solution\ExecuteTutorial.vbp
'*****
'***** Inputs:
'*****           Schedule File - Contains the Moniker used to
'*****                           launch the schedule
'*****           Data File - Contains the location of the
'*****                           XML document to be passed to
'*****                           the schedule for processing.
'*****
'***** Outputs:
'*****           Data File - Data file is passed to MSMQ
'*****                           for later retrieval by the schedule.

Private g_MSMTxDisp As MSMQ.MSMQTransactionDispenser
Private g_MSMQQueue As MSMQ.MSMQQueue
Private g_MSMQInfo As MSMQ.MSMQQueueInfo
Private g_CurSkedDir As String
Private g_CurDataDir As String

Private Sub Class_Initialize()
    Set g_MSMQInfo = CreateObject("MSMQ.MSMQQueueInfo")
    Set g_MSMTxDisp = CreateObject("MSMQ.MSMQTransactionDispenser")
End Sub

Public Sub RunSchedule(ByVal strScheduleFile As String, ByVal strData As String)
    Dim objfs As New FileSystemObject
    On Error GoTo cmdRunSked_Click_err

    'Connect To MSMQ and Remove Any Existing Messages
```

```

PurgeMSMQ "DIRECT=OS:.\private$\ReceivePoReq"

'Send Selected message to MSMQ
ExecuteMSMQ "DIRECT=OS:.\private$\ReceivePoReq", strData

'Start Schedule which reads message from MSMQ
ExecuteSchedule strScheduleFile

Exit Sub

cmdRunSked_Click_err:
    MsgBox Err.Description & vbCrLf & "Error: " & Err.Number & " (0x" &
Hex(Err.Number) & ")", vbCritical, "Error " & Err.Source
    Err.Clear

End Sub

Private Sub PurgeMSMQ(ByVal strQueuePath As String)
    Dim l_MSMQMsg As MSMQMessage

    On Error GoTo Err_ConnectMSMQ
    g_MSMQInfo.FormatName = strQueuePath
    Set g_MSMQQueue = g_MSMQInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)

    On Error GoTo Err_PurgeMSMQ
    Do
        Set l_MSMQMsg = g_MSMQQueue.Receive(, , , 1)
    Loop While Not l_MSMQMsg Is Nothing
    Exit Sub

Err_ConnectMSMQ:
    Err.Raise Err.Number, "Connecting To MSMQ", "Could Not Open the MSMQ Queue ""
& strQueuePath & ""."" & vbCrLf & vbCrLf & Err.Description
    Exit Sub
Err_PurgeMSMQ:
    Err.Raise Err.Number, "Cleaning MSMQ", "Could Not Remove Existing Messages from
MSMQ Queue "" & strQueuePath & ""."" & vbCrLf & vbCrLf & Err.Description
    Exit Sub
End Sub

Private Sub ExecuteMSMQ(ByVal strQueuePath As String, DataToQueue As String)
    Dim QueueMsg As New MSMQMessage

    Dim strData As String
    Dim fSend As Boolean
    Dim txt As TextStream
    Dim mybyte() As Byte

    On Error GoTo Err_SendMSMQ
    g_MSMQInfo.FormatName = strQueuePath
    Set g_MSMQQueue = g_MSMQInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
    mybyte = StrConv(DataToQueue, vbFromUnicode)
    QueueMsg.Body = DataToQueue

```

```

    Dim MSMQTx As Object
    Set MSMQTx = g_MSMTxDisp.BeginTransaction
    QueueMsg.send g_MSMTxQueue, MSMQTx
    MSMQTx.Commit

    Set QueueMsg = Nothing
    Set MSMQTx = Nothing
    Exit Sub

Err_SendMSMQ:
    Err.Raise Err.Number, "Sending Message To MSMQ", "Could Not Send Message To
    MSMQ Queue "" & strQueuePath & ""."" & vbCrLf & vbCrLf & Err.Description
    Exit Sub
End Sub

Private Sub ExecuteSchedule(ByVal strSchedule)
    Dim SendPAQ As Object
    On Error GoTo Err_ExecSched

    Set SendPAQ = GetObject(strSchedule)
    If SendPAQ Is Nothing Then
        Err.Raise vbObjectError + 1, , "Invalid Schedule Handle Returned."
    End If
    Set SendPAQ = Nothing
    Exit Sub

Err_ExecSched:
    Err.Raise Err.Number, "Starting Schedule", "Could Not Launch the XLANG
    Schedule" & vbCrLf & "Please verify the path to the SKX file and path to your data
    are correct. Also make sure the private queues have been created." & vbCrLf &
    vbCrLf & Err.Description
    Exit Sub
End Sub

```

Java Interoperability Solution

The PeopleSoft Java interoperability solution allows you to write Java applications that interact with the PeopleSoft EnterpriseOne system. The Java interoperability solution includes the following types of connectors:

- Dynamic Java Connector
- Java Connector
- Java Connector Architecture (JCA) Resource Adapter

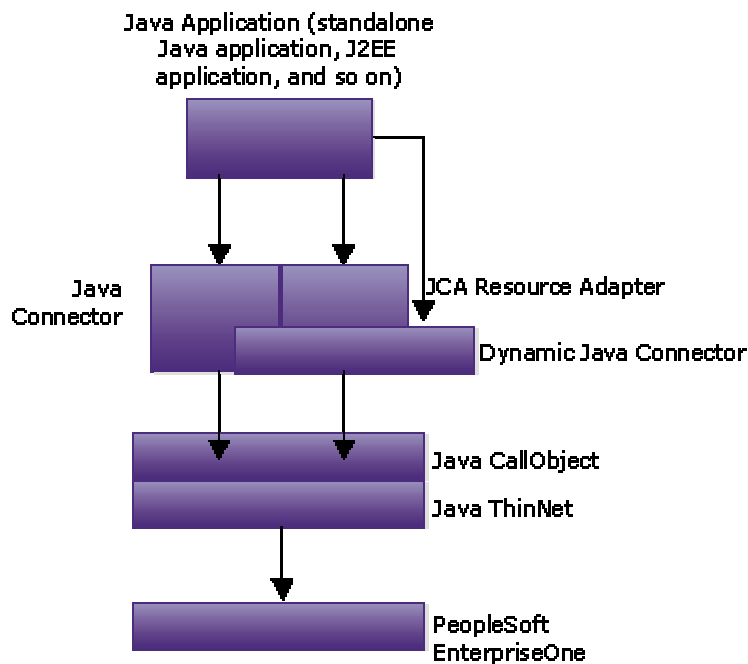
The initial Java interoperability solution provided by PeopleSoft is the Java connector. The Java connector generates a Java wrapper object around the PeopleSoft EnterpriseOne business function and data structure. A Java application calls the business functions from the Java wrapper object.

The dynamic Java connector is an enhancement to the Java connector. The dynamic Java connector allows Java applications to dynamically call business function without generating business function wrappers. The dynamic Java connector ensures that the Java business function is compatible with the server spec. The dynamic Java connector makes it much easier for the Java application to switch between PeopleSoft EnterpriseOne environments.

The JCA resource adapter is a thin layer built on top of the dynamic Java connector that provides standard APIs required by the Java connector architecture. The core functionality for the JCA resource adapter is to interact with PeopleSoft EnterpriseOne, and this functionality is leveraged to the dynamic Java connector.

Each connector has a complete set of APIs that allow Java applications to interact with PeopleSoft EnterpriseOne. You can choose the type of connector that best satisfies your purposes.

The following diagram shows how a Java application interacts with PeopleSoft EnterpriseOne through a connector:



Generally, each connector provides public interfaces (or APIs) for the following services that can be used by a Java application:

Service	Description
Security Management	Handle security access to the PeopleSoft EnterpriseOne system.
User Session Management	Manage the user session pooling.
Business Function Calls	How the Java application calls business functions.
Transaction Management	Manage the transaction process to the PeopleSoft EnterpriseOne system.
Error Handling	Provide the appropriate exceptions to the connector user to easily handle error scenarios.

Both the Java connector and the dynamic Java connector support the processing of outbound events.

Note

If this is your first implementation of a Java connector, PeopleSoft suggests you consider the dynamic Java connector instead of the Java connector. The functional capabilities are the same. The advantage of implementing the dynamic Java connector is that you are not required to generate wrappers.

Dynamic Java Connector and PeopleSoft EnterpriseOne

The dynamic Java connector allows a Java application to call a business function. Compared to the Java connector, the dynamic Java connector has the following distinguishing features:

- Dynamically introspects business function metadata. The business function metadata is introspected from the PeopleSoft EnterpriseOne server during application design time by using connector APIs without pre-generating business function wrappers.
- Dynamically calls business functions without pre-generating business function wrappers. Since there is no local storage of business function spec metadata, the business function used by the dynamic Java connector is always compatible with the server spec metadata.
- Easily switches from one environment to another environment. The Java application can run on any environment that is compatible to the environment on which the Java application was designed.

The dynamic Java connector provides the following services:

- For application design, it allows client programs to introspect business function specification metadata.
- For application deployment, it validates whether a client application can run through a certain PeopleSoft EnterpriseOne server.
- For application runtime, it provides an interface that allows the connector client to call the business function on the PeopleSoft EnterpriseOne server.

Each server is described in detail in corresponding sections of this guide.

Design Time for the Dynamic Java Connector

This section covers considerations for designing your dynamic Java connector.

Business Function Spec Metadata Introspection

To call a business function method, you need to know the business function methods that are available to be called and you need to know about the business function metadata. The following list provides examples of metadata:

- Business function method (such as F4211BeginDoc)
- The module name (C file name) to which a business function method belongs (such as B123456)
- Description of the business function method (such as sales order)
- Data structure template name that is associated with a business function method (such as D123456)
- The attributes for all of the data items (parameters) in a business function method, such as name=szMnAddressbookNumber, itemID=1, data type=Math_Numeric, length=48, requiredType="Yes", IOType="INOUT"

In the dynamic Java connector, metadata is represented by the BSFNMethod and BSFNParameter interfaces.

BSFNMethod

The BSFNMethod interface defines APIs that allow you to retrieve metadata related to the business function method. The BSFNMethod interface defines the following APIs:

- `public String getName();`
- `public String getDSTemplateName();`
- `public String getBSFNName();`
- `public String getDescription();`
- `public BSFNParameter getParamger(String paraName);`
- `public BSFNParameter[] getParameters();`
- `public String getFormatString();`
- `public ExecutableMethod createExecutable();`
- `public boolean equals(Object anotherBSFNMethod);`
- `public void setEqualTo(BSFNMethod anotherBSFNMethod);`
- `public String getVersion();`
- `public void setVersion(String version);`

BSFNParameter

The BSFNParameter interface defines APIs that allow you to retrieve metadata related to the data structure of the business function. The BSFNParameter interface defines the following APIs:

- `public int getItemID();`
- `public String getName();`
- `public int getLength();`
- `public IOType getIOType();`
- `public RequiredType getRequiredType();`
- `public BSFNDataType get DataType();`

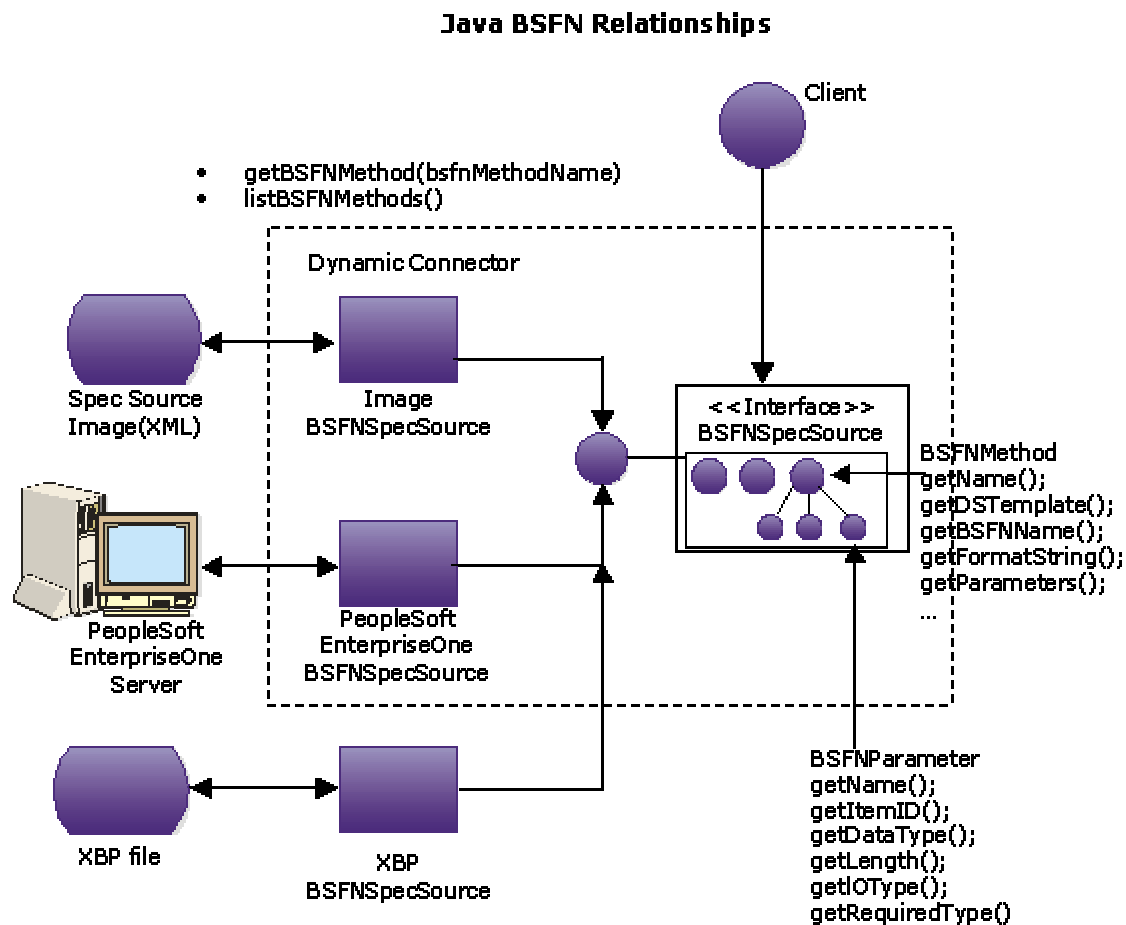
BSFNSpecSource

You can write a program to retrieve business function method metadata through an interface called BSFNSpecSource. The BSFNSpecSource interface defines the following APIs:

- `Public BSFNMethod getBSFNMethod(String methodName) throws SpecFailureException`
- `Public BSFNMethod[] getBSFNMethods() throws SpecFailureException`

The class that implements the BSFNSpecSource interface reads the business function method metadata from an external physical repository and creates the BSFNMethod object. AbstractBSFNSpecSource is an abstract implementation of BSFNSpecSource provided by the dynamic Java connector. All customized implementations of BSFNSpecSource should be a subclass of this class. OneWorldBSFNSpecSource is the default implementation of AbstractBSFNSpecSource.

The following illustration shows the BSFNSpecSource, BSFNMethod, and BSFNParameter relationships:



The following example of code shows how to retrieve the BSFN spec from BSFNSpecSource:

```
//Step 1: Create a new BSFNSpecSource
BSFNSpecSource specSource = null;
int sessionID = Connector.getInstance().login("user", "pwd", "env", "role");
specSource = new OneWorldBSFNSpecSource(sessionID);
// or specSource = new ImageBSFNSpecSource("SSI.xml");

//Step 2: Get BSFNMethod by name from specSource
BSFNMethod method = specSource.getBSFNMethod("GetEffectiveAddress");

//Step 3: Introspect BSFNMethod metadata
method.getName();
...
BSFNParameter[] paraList = method.getParameters();
for (int I=0; I<paraList.length;i++) {
    BSFNParameter para = paraList[i];
    para.getName();
    para.getDataType();
}
...
}
```

SpecDictionary

A BSFNSpecSource can contain thousands of business function methods. The dynamic Java connector provides an interface to properly categorize and organize business function methods. Without proper categorization and organization, it is difficult to navigate and find the proper business function method. To solve this problem, the dynamic Java connector provides an interface called SpecDictionary, which provides the following services:

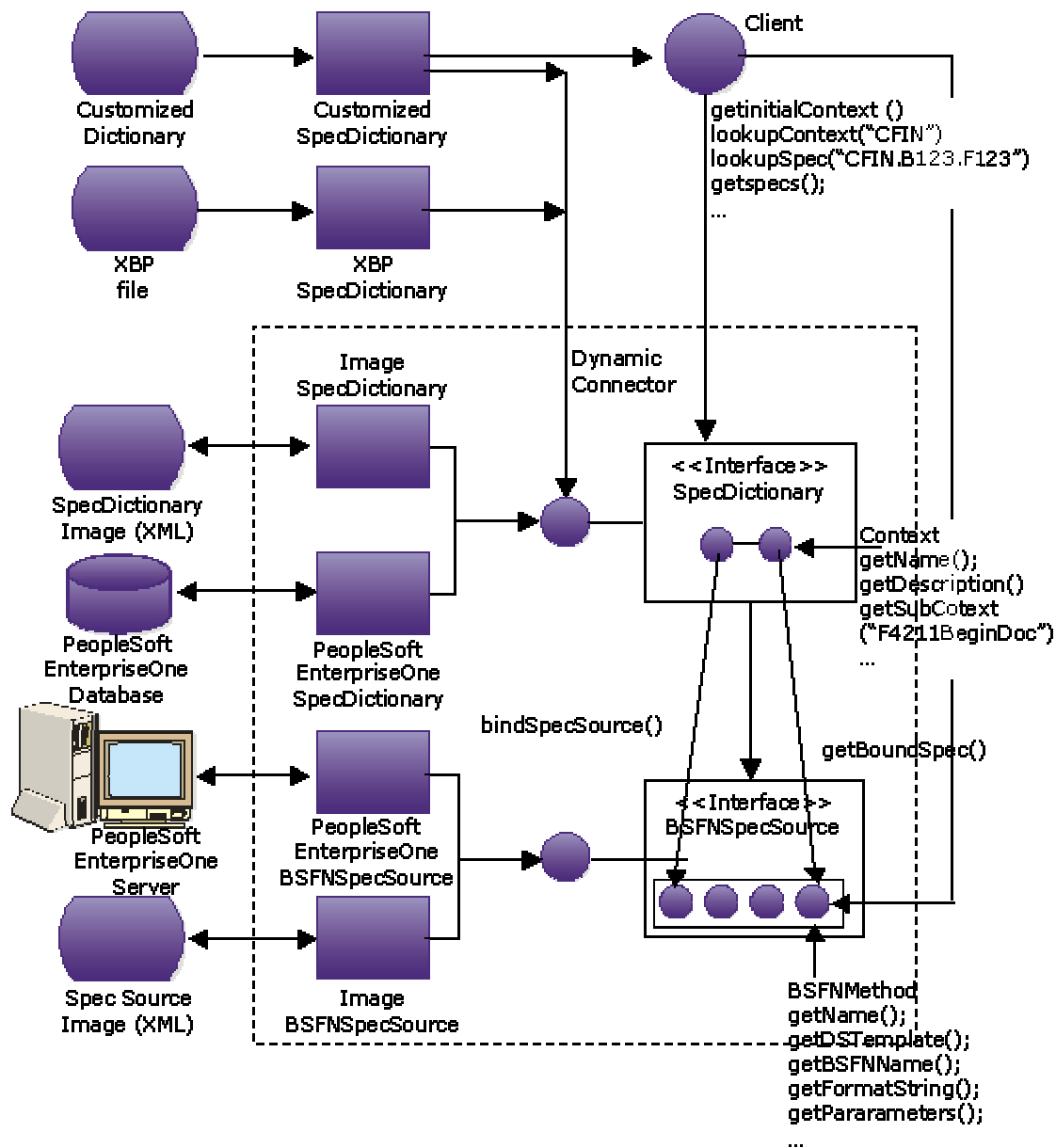
- Categorizes business function methods in a hierarchy
- Masks the BSFNSpecSource and limits the number of business function methods a client can view

The entry of SpecDictionary is called a context. A context is a set of name-to-object bindings. Every context has an associated naming convention. A context provides a lookup operation that returns the object. The dynamic Java connector provides the following two concrete classes that implement the SpecDictionary:

- OneWorldSpecDictionary, which gets the hierarchy information from the PeopleSoft EnterpriseOne database. OneWorldSpecDictionary categorizes business function methods as DLL library – C file name – C function name.
- ImagespecDictionary, which gets the hierarchy information from Spec Dictionary Image, which is an XML file.

Like BSFNSpecSource, third-party programs can store the spec dictionary information in their proprietary format, but they need to implement their own specDictionary to read the proprietary spec.

The following diagram shows the relationship between SpecDictionary and BSFNSpecSource:



The following example code shows how to use SpecDictionary and BSFNSpecSource to browse and lookup information:

```
BSFNSpecSource specSource = null;
SpecDictionary specDictionary = null;
//Step 1: Create a SpecDictionary
int sessionID = Connector.getInstance().login("user", "pwd", "env", "role");
specDictionary = new OneworldSpecDictionary(sessionID);
// or   specDictionary = new ImagespecDictionary("dict.xml");

//Step 2: Bind the SpecDictionary to a SpecSource
specDictionary.bindSpecSource(specSource);

//Step 3a: Lookup the BSFNMethod by giving the full path
BSFNMethod method =(BSFNMethod) specDictionary.getSpec
("CFIN.F4211.F4211BeginDoc");

//Step 3b: or navigate through the dictionary and get the context attributes
Context initContext = specDictionary.getInitialContext();
Context[] subContextList = initContext.getSubcontexts();
for (int I=0;I<subContextList.length; I++) {
    Context subContext=subContextList[i];
    subContext.getName();
    subContext.getDescription();
    method=(BSFNMethod) subContext.getBoundSpecContent();
    ...
}
```

Validate Business Function Spec Metadata

If your dynamic Java connector program calls a business function from OneWorldBSFNSpecSource, you do not need to validate the business function metadata. The business function metadata in OneWorldBSFNSpecSource is always the same as the business function metadata that is on the PeopleSoft EnterpriseOne server where the business function runs. You must ensure that all input parameters are set correctly, according to OneWorldBSFNSpecSource.

If your dynamic Java connector program calls a business function from a spec source other than OneWorldBSFNSpecSource (such as ImageBSFNSpecSource or a custom business function spec source), the business function metadata that is in your local spec source might not be compatible with the business function metadata that is on the PeopleSoft EnterpriseOne server where the business function will run. Local business function spec metadata can be validated during the following conditions:

Deploy Time	Your dynamic Java connector program validates your local spec source against the PeopleSoft EnterpriseOne server spec source before run time. PeopleSoft recommends this validation, as all business functions in your local spec source are validated, and your program can be redesigned before it is shipped.
Run Time	The dynamic Java connector validates your program based on the local spec design when running business functions. During this condition, only the business function that is called is validated. Run time validations should be treated as error handling when incompatible business function specs are found.

The dynamic Java connector supplies two ways to validate business function spec metadata during deploy time: SpecImageValidator APIs and SpecImageConsole command line.

The APIs for SpecImageValidator are:

- `public SpecImageValidator(BSFNSpecSource srcSpecSource)`
- `public ValidationResultSet validate(SpecDictionary dictionary) throws SpecFailureException`
- `public ValidationResultSet validate(SpecDictionary dictionary, String path) throws SpecFailureException`
- `public ValidationResultSet validate(BSFNSpecSource dstSpecSource) throws SpecFailureException`
- `public ValidationResultSet validate(BSFNSpecSource dstSpecSource, String bsfnMethodName)`

Note

If the SpecImageConsole command line is used, the dynamic Java connector can only validate business function spec metadata from ImageBSFNSpecSource; custom business function spec sources cannot be validated.

Validating business function metadata from the SpecImageConsole command line is discussed later in this section.

Using the SpecImageConsole

You can use the SpecImageConsole to generate, update, validate and synchronize spec images.

Generate Spec Image

You use the spec image console to generate or regenerate a spec image. The following information is useful for generating or regenerating a spec image.

Usage

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole /Generate [Other Options]
```

Options

`/UserName <user> (required)`

`/Password <pwd> (required)`

`/Env <environmet> (required)`

`/Role <role> (required)`

`/ImageStub <stub file> (required)`

`/ImageType <image type [SSI|SDI|ALL]> (optional, default is ALL)`

/ErrorFile <error file> (optional, default is System.err)

/OutputFile <output file> (optional, default is System.out)

Explanation

Log on to PeopleSoft EnterpriseOne with <user>, <pwd>, <environment>, and <role>.

Load the spec image stub from <stub file>.

Generate the spec image with the image type <image type>.

The spec image will be written to the <output file> (or System.out if /OutputFile not present).

Error messages will be written to the <error file> (or System.err if /ErrorFile not present).

Example

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole  
/Generate /ImageStub image_stub.xml /ImageType SDI /OutputFile  
image.xml /ErrorFile err.log
```

Update Spec Image

You can use the spec image console to update or change a spec image. The following information is useful for updating a spec image.

Usage

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole /Update [Other  
Options]
```

Options

/UserName <user> (required)

/Password <pwd> (required)

/Env <environmet> (required)

/Role <role> (required)

/SSI <SSI file> (required)

/SDI <SDI file> (optional)

/AddSpec <BSFNSpec name> (for example, F4211BeginDoc; optional)

/AddContext <full Context name> (for example, CFIN.B3100010 or
CFIN.B3100010.F4211BeginDoc; optional)

/RemoveSpec <BSFNSpec name> (for example, F4211BeginDoc; optional)

/RemoveContext <full Context name> (for example, CFIN.B3100010 or
CFIN.B3100010.F4211BeginDoc; optional)

Explanation

Log on to PeopleSoft EnterpriseOne with <user>, <pwd>, <environment>, and <role>.

Load the <SDI file> (If option /SDI not present, then load <SSI file>) add/remove the context and BSFN spec that is specified as <full Context name> and <BSFNSpec name>.

Example

The following example shows how to update the Spec Dictionary Image (sdi.xml) and the Spec Content Image (SSI.xml). The example adds Context CFIN.B00100, removes Context CFIN.B001002, adds Spec F4211BeginDoc, and removes Spec F4311BeginDoc.

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole
/Update /SDI sdi.xml /SSI ssi.xml /addContext CFIN.B001001
/removeContext CFIN.B001002 /addSpec F4211BeginDoc /removeSpec
F4311BeginDoc
```

Validate Spec Image

You can use the spec image console to validate the spec image against the PeopleSoft EnterpriseOne server. The following information is useful for validating a spec image.

Usage

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole /Validate [Other
Options]
```

Options

/UserName <user> (required)

/Password <pwd> (required)

/Env <environment> (required)

/Role <role> (required)

/SSI <SSI file> (required)

/SDI <SDI file> (optional)

/OutputFile (optional, default to System.out)

Explanation

Log on to PeopleSoft EnterpriseOne with <user>, <pwd>, <environment>, and <role>.

If option /SDI is present, validate all the BSFNSpec that bind to the <SDI file>. If /SDI is not present, validate all the BSFNSpec in the <SSI file>.

The spec image will be written to the <output file> (or System.out if /OutputFile is not present).

Example

The following example shows how to validate spec image using ssi.xml as the SpecDictionary and sdi.xml as the SpecSource. The example writes the validation result to validateResult.log.

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole  
/Validate /SDI sdi.xml /SSI ssi.xml /OutputFile validateResult.log
```

Synchronize Spec Image

You can use the spec image console to synchronize the spec image with the PeopleSoft EnterpriseOne server. The following information is useful for validating a spec image.

Usage

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole /Synchronize [Other  
Options]
```

Options

```
/UserName <user> (required)  
  
/Password <pwd> (required)  
  
/Env <environment> (required)  
  
/Role <role> (required)  
  
/SSI <SSI file> (required)  
  
/SDI <SDI file> (optional)  
  
/ErrorFile <err file>(optional, default to System.err)
```

Explanation

Log on to PeopleSoft EnterpriseOne with <user>, <pwd>, <environment>, and <role>.

If option /SDI present, synchronize all the BSFNSpec that bind to the <SDI file>. If /SDI is not present, synchronize all the BSFNSpec in the <SSI file>.

The new spec image will be written to the <SSI file>. Error messages will be written to <err file> (or System.err if /ErrorFile is not present).

Example

The following example shows how to synchronize the spec source image, ssi.xml:

```
java com.jdedwards.system.connector.dynamic.util.SpecImageConsole  
/Synchronize /SSI ssi.xml
```

Installing Dynamic Java Connector on a Non-PeopleSoft EnterpriseOne Environment

The following steps show how to install dynamic connector components so that you can run a dynamic Java connector application on a third-party machine.

1. Copy the following files from the PeopleSoft EnterpriseOne server to a directory on the non-PeopleSoft EnterpriseOne machine that you want to use:

- Connector.jar
- kernel.jar
- jdeutil.jar
- database.jar
- log4j.jar
- xerces.jar
- jdeinterop.ini
- jdelog.properties
- JDBC driver(s) (you need to get the JDBC driver(s) from your database vendor)

For example, you might copy the above files to the following directory on your non-PeopleSoft EnterpriseOne machine:

C:\PeopleSoft\Interop

2. Add the following files to the CLASSPATH:

- Connector.jar
- kernel.jar
- jdeutil.jar
- database.jar
- log4j.jar
- xerces.jar
- JDBC driver(s)

3. Add the path where the jdelog.properties and jdeinterop.ini files are located into CLASSPATH.
4. Edit jdeinterop.ini and jdelog.properties for proper settings.

See Also

- *jdeinterop.ini* in the *Connectors Guide* for more information about the jdeinterop.ini sections and settings

Runtime for the Dynamic Java Connector

This section covers runtime considerations for your dynamic Java connector.

Call a Business Function

If you know the business function name and the parameters (data items) associated with the business function, you can use the dynamic Java connector to call the business function. Unlike previous versions of the Java connector, the dynamic Java connector does not require pre-generated wrappers. The following code sample shows you how to call a business function using the dynamic Java connector.

```
// Step 1: Login
int sessionID = Connector.getInstance().login("user", "pwd", "env", "role");
// Pre-condition: create the SpecDictionary or BSFNSpecSource as mentioned before
// Step 2: Lookup the bsfn method from SpecDictionary or BSFNSpecSource as
// mentioned before
BSFNMethod bsfnMethod = (BSFNMethod)specSource.getBSFNMethod ("Get
EffectiveAddress");
// Step 3: create the executable method from the BSFN metadata
ExecutableMethod addressbook = bsfnMethod.createExecutable();

try {
// Step 4: Set parameter values
    addressbook.setValue("mnAddressNumber", "1");
// Step 5: Execute the business function
    BSFNExecutionWarning warning = addressbook.execute(sessionID);
// Step 6: Get return parameter values

    System.out.println("szNamealpha=" + addressbook.getValueString
("szNamealpha"));
// Get the warnings if any
    if (warning.hasWarnings()){
        String warningMsgs[] = warning.getWarningMessages();
        for (int i=0;i<warningMsgs.length;i++){
            System.out.println(warningMsgs[i]);
        }
    }
} catch (SystemException e) {
    //SystemException is thrown when system crash, this is fatal
    // error and must be caught
    System.exit(1);
} catch (ApplicationException e){
    // ApplicationException is thrown when business function
    // execution fail, this is RuntimeException and thus can be
    // unchecked. But it is strongly recommend to catch this
    // exception
} finally {
//Log off and shut down connector if necessary
connector.logoff(sessionID);
connector.shutdown();
}
```

The dynamic Java connector allows you to use hash tables to input parameter values. The following example code illustrates inputting parameter values using the Hashtable class.

```
Map input = new Hashtable();
input.put("mnAddressNumber", String.valueOf(addressNo));
addressbook.setValues(input);
```

The dynamic Java connector allows you to use hash tables to retrieve output values. The following example code illustrates retrieving output values using the Hashtable class.

```
Map output = addressbook.getValues();
System.out.println("szNamealpha=" + output.getValueString("szNamealpha"));
```

BSFN Cache

The dynamic Java connector fetches a business function spec from a SpecSource (PeopleSoft EnterpriseOne server or an XML repository) to create an executable method. To reduce some of the overhead for creating executable methods during run business functions, the Java connector caches the executable methods after they are created.

If OneWorldSpecSource is used as SpecSource, the dynamic Java connector gets the most current business function spec from the PeopleSoft EnterpriseOne server the first time the business function is called. The cache is destructed after the connector is shutdown. This cache mechanism expedites business function execution by eliminating the overhead of fetching the business function spec for every business function call.

The duration of the cache can be configured in the jdeinterop.ini file. You can configure the setting to balance the speed of the business function execution and the update of the business function Spec.

Transaction via Dynamic Java Connector

You can use the dynamic Java connector to do a PeopleSoft EnterpriseOne transaction in either auto or manual mode. The following example code for a purchase order entry transaction shows the steps for using the dynamic Java connector in manual mode.

```
int sessionID = Connector.getInstance().login("user", "pwd", "env", "role");
UserSession userSession = Connector.getInstance().getUserSession (sessionID);
boolean isManualCommit;
//set isManualCommit as true or false

//Step 1: create OneWorldTransaction
OneWorldTransaction transaction = userSession.createOneWorldTransaction
(isManualCommit);

// Step2: create the Purchase Order Entry executable methods (such as
// poeBeginDoc, poeEditLine, poeEndDoc) from the BSFN metadata as //mentioned
before

//Step 3: begin the transaction
transaction.begin();

//Step 4: run BSFNs in this transaction
//set poeBeginDoc input parameters as mentioned above, code not shown
BSFNExecutionWarning warning = poeBeginDoc.execute(transaction);
//set poeEditLine input parameters as mentioned above, code not shown
BSFNExecutionWarning warning = poeEditLine.execute(transaction);
//set poeEndDocinput parameters as mentioned above, code not shown
BSFNExecutionWarning warning = poeEndDoc.execute(transaction);

//Step 5: Commit or rollback transaction
transaction.commit();
//or transaction.rollback();
```

OCM Support for the Dynamic Java Connector

You use Object Configuration Manager (OCM) to map business functions to a PeopleSoft EnterpriseOne server so that the Dynamic Java connector can access OCM to run business functions. You no longer configure the jdeinterop.ini file to define the PeopleSoft EnterpriseOne from which you want to execute business functions. Using OCM support should result in an increase in performance, scalability, and load balancing. The Java interoperability server distributes the processes of the Java client to various PeopleSoft EnterpriseOne servers depending on user, environment, and role. To take advantage of Dynamic Java connector OCM support, you must do the following:

- Configure your OCM and map your business function on different PeopleSoft EnterpriseOne servers
- Set OCMEEnabled=true in jdeinterop.ini
- Configure the settings in jdeinterop.ini regarding the bootstrap data source with your OCM configuration

Ensure that the following settings in your jdeinterop.ini configuration file are set:

jdeinterop.ini File Section	Required Settings
OCM	OCMEnabled
JDBj-BOOTSTRAP SESSION	user, password, environment, and role
JDBj-BOOTSTRAP DATA SOURCE	name, databaseType, server, database, serverPort, physicalDatabase, library, owner
[JDBj-JDBC DRIVERS]	ORACLE, iSeries, SQLSERVER, UDB
[JDBj-ORACLE]	tns

User Session Management for the Dynamic Java Connector

When the connector user successfully signs on, a valid user session is allocated to that user signon. The user session has status for two types of connector operations, one is for inbound business function calls, and the other is for outbound real-time events. The connector monitors the status of the user session, and uses the time out settings in the jdeinterop.ini file to stop the user session when a time out setting has been reached. The connector looks at the following settings:

jdeinterop.ini File Section	Setting	Explanation
[CACHE]	UserSession	The maximum connector idle time for an inbound business function call.
[INTEROP]	manual_timeout	The maximum idle time for a manual transaction.
[EVENTS]	outbound_timeout	The maximum value of connector idle time for receiving outbound events.

The values for the settings are in milliseconds. A value of zero (0) indicates infinite time out. The above settings are defined in the jdeinterop.ini section of this guide.

If an inbound user session times out, that user session cannot be used to execute a business function call. Likewise, if an outbound user session times out, that user session cannot be used for events. When both, inbound and outbound, sessions time out, the user session is removed from the connector. Since each user session has a corresponding handle in the PeopleSoft EnterpriseOne server, PeopleSoft *highly* recommends that you explicitly call a connector API to log off the user session. The API log off releases the handle in the PeopleSoft EnterpriseOne server when the user session is no longer used.

The following sample code shows how to retrieve and manage a user session:

```
// Login
int sessionId = Connector.getInstance().login("user", "pwd", "env", "role");

// Do whatever using the sessionId. If InvalidSessionException is caught, user
session is not valid any more

//Check the status of the usersession
UserSession session;
try{
    session=Connector.getInstance().getUserSession(sessionId);
}catch(InvalidSessionException ex){
    System.out.println("Invalid user session");
}
if(session.isInboundTimedout()){
    System.out.println("User session inbound is timed out");
}
if(session.isOutboundTimedout()){
    System.out.println("User session outbound is timed out");
}

//Log off and shut down connector to release user session from the server
connector.logoff(sessionId);
connector.shutdown();
```

Inbound XML Request via Dynamic Java Connector

You can use the dynamic Java connector to send inbound synchronous XML requests (such as XML CallObject and XML List) to the PeopleSoft EnterpriseOne server.

The following example code shows how to use the dynamic Java connector to execute an inbound XML request:

```
String xmlDoc;
//or byte[] xmlDoc
//...Load a String or byte[] into xmlDoc;

String requestResult;
try {
    XMLRequest xmlRequest = new XMLRequest(hostname, port, xmlDoc);
    requestResult = xmlRequest.execute();
} catch (IOException e) {
    System.out.println("Error in XML request");
}
//...handle requestResult.
```

See Also

See the following topics in the *Interoperability Guide*:

- ❑ *XML CallObject* for more information about using XML CallObject
- ❑ *XML List* for more information about using XML List
- ❑ *XML Transaction* for more information about using XML Transaction

Logging for the Dynamic Java Connector

Dynamic Java connector logging is built on top of Apache Open Source Project Log4j. Log4j supports five levels of logging, which are listed below in order of severity, from less to more:

- DEBUG
- INFO
- WARNING
- ERROR
- FATAL

The dynamic Java connector provides the following APIs, located in `ConnectorLog.java`, to support logging information:

- `public static void debug(Object source)`
- `public static void info(Object source)`
- `public static void warn(Object source)`
- `public static void warn(Object source, Throwable err)`
- `public static void error(Object source, Throwable err)`
- `public static void error(Object source)`
- `public static void fatal(Object source)`
- `public static void fatal(Object source, Throwable err)`

Log properties (such as log file location, level of log messages to show in log file, and so on) are set in `jdelog.properties`. The `jdelog.properties` settings provide flexibility for dynamic Java connector applications to log messages according to your needs. For example, you might set log level to ERROR or FATAL for a production environment or to DEBUG for a development or test environment.

See Also

- ❑ *jdelog.properties* in the *Connectors Guide* for more information about setting up logging configurations

Exception Handling for the Dynamic Java Connector

The dynamic Java connector error handling design provides flexibility for you to decide how to handle application-level errors. The dynamic Java connector provides the following two types of exceptions to handle errors:

- **ApplicationException:** This is the super class of all exceptions that result from application errors, such as `InvalidConfigurationException` (invalid INI settings), `InvalidLoginException` (invalid login), `InvalidDataTypeException` (invalid BSFN data type), and so on. The `ApplicationException` is a runtime exception. It is up to the client program to catch this type of exception.
- **SystemException:** This is the super class of all exceptions that result from system errors, such as `ServerFailureException` (server down or connection failure), `BSFNLookupFailureException` (unable to find BSFN information in PeopleSoft EnterpriseOne tables), and `SpecFailureException` (unable to connect to Spec Source). It is up to the client program to catch this type of exception.

Sample Applications

The following applications are shipped with the dynamic Java connector in their Java source form:

Address Book	Queries an AddressBook entry
Events	Subscribes to events
Manual Commit	Performs a local transaction using a Purchase Order Entry application
Purchase Order	Enters a purchase order
Sales Order	Enters a sales order

Prerequisites

- ❑ Create a directory for the sample applications (for example, `C:\connectorsamples`).
- ❑ Install a Java Development Kit (JDK) version 1.3 or higher. Be sure to install a full JDK and not the Java Runtime Environment (JRE).
- ❑ Set the `JAVA_HOME` environment variable to the JDK parent directory.
- ❑ Set up in the `jdeinterop.ini` and `jdelog.properties` files and place the files in the directory you created for the sample applications (for example, `C:\connectorsamples`).

Note

You can download the JDK from the Sun Microsystems website (java.sun.com/j2se).

For more information about installing the JDK, see *Installing Dynamic Java Connector on a Non-PeopleSoft EnterpriseOne Environment* in the *Connectors Guide*.

For more information about setting the `jdeinterop.ini` file, see *jdeinterop.ini* in the *Connectors Guide*.

For more information about setting the `jdelog.properties` file, see *jdelog.properties* in the *Connectors Guide*.

Compile the Sample Applications

The sample applications are shipped in their Java source form, which allows you to see the usage of the dynamic Java connector API. You must compile these sample applications in your environment before you can run them. Use the following steps to compile the sample applications:

- Locate the `connector_samples_src.jar` file. This file is on the PeopleSoft EnterpriseOne Java Server CD, under the `system/classes/samples` directory:
- Unzip the entire contents of the `connector_samples_src.jar` file into the directory you created (for example, `C:\connectorsamples`). The `.jar` file is a traditional `.zip` file with the Java `.jar` extension. The `.jar` file contains all of the sample application source files (`.java` files). All of the `.jar` files that you need for both compiling and running the sample applications are in the `system/classes` directory on the PeopleSoft EnterpriseOne Java Server CD.
- Compile the samples. To compile the samples, you need the `Connector.jar` and the `kernel.jar` files. The following compile script assumes that you are in the parent directory where you extracted the sample application `.jar` file contents, and that you have copied the necessary `.jar` files to that parent directory. The compile script assumes that you are going to compile all of the applications at once. Enter the following compile script as a single, continuous line in your command prompt window.

```
javac -classpath ../Connector.jar;../kernel.jar
com/jdedwards/system/connector/dynamic/sample/*.java
com/jdedwards/system/connector/dynamic/sample/addressbook/*.java
com/jdedwards/system/connector/dynamic/sample/events/*.java
com/jdedwards/system/connector/dynamic/sample/manualCommit/*.java
com/jdedwards/system/connector/dynamic/sample/purchaseorder/*.java
com/jdedwards/system/connector/dynamic/sample/salesorder/*.java
com/jdedwards/system/connector/dynamic/sample/speconsole/*.java
com/jdedwards/system/connector/ sample/utctime/*.java
com/jdedwards/system/connector/dynamic/sample/util/*.java
```

Run the Sample Applications

After you compile the sample applications, you can run them. The .jar files that are used in the following script, and the jdeinterop.ini and jdelog.properties files, must be in the directory that you created (C:\connectorsamples). To run any of the sample applications, enter the following script in a command prompt window in the C:\connectorsamples directory:

```
Java
-classpath ../Connector.jar;../database.jar;../jdeutil.jar; ../kernel.jar;
../log4j.jar;../xerces.jar; (JDBC driver .jar files); (path to jdeinterop.ini and
jdelog.properties files) (sample application main class name)
```

Execute this script as a single, continuous command. All of the .jar files listed after the "-classpath" argument must be kept together with no spaces (except after the "-classpath" word itself). The sample application main class name for each sample application is listed in the following table. For several of the applications, a dialog box (requesting your PeopleSoft EnterpriseOne credentials) appears. Be sure to use all capital letters when you enter your credentials.

Sample Application	Main Class Name
Address Book	com.jdedwards.system.connector.dynamic.sample.addressbook.AddressbookClient
Events	com.jdedwards.system.connector.dynamic.sample.events.SinkFrame
Manual Commit	com.jdedwards.system.connector.dynamic.sample.manualCommit.PurchaseOrderEntryClient
Purchase Order	com.jdedwards.system.connector.dynamic.sample.purchaseorder.PurchaseOrderEntryClient
Sales Order	com.jdedwards.system.connector.dynamic.sample.salesorder.SalesOrderEntryClient

Java Connector and PeopleSoft EnterpriseOne

A business function is a logical collection of C functions and their associated data structures grouped together to produce a unit of work. PeopleSoft EnterpriseOne Java objects are wrappers, implemented in Java, around these business functions and data structures.

The method that a Java wrapper provides has a one-to-one correspondence with business functions. Because all methods must be defined in a Java class, a library must be defined in the corresponding iJDEScript file.

For example, if library A contains business function B550001, and within this business function two C functions exist, named foo1 and foo2, with data structures for each function named DS1 and DS2, then the corresponding Java class would be as follows:

```
Public class A
{
    public int foo1(DS 1 param, OneWorldInterface ow,
        Connector c, int handle)
    {
        0
    }
    public int foo2(DS2 param, OneWorldInterface ow,
        Connector c, int handle)
    {
        0
    }
    public DS1 Createfoo1ParameterSet()
    {
        0
    }
    public DS2 Createfoo2ParameterSet()
    {
        0
    }
    0
}
```

For each business function X, a method CreateXParameterSet() exists in the class that returns a class for the data structure used by the business function.

Each data structure has a corresponding Java class, and each element in the data structure has a *get* and a *set* method. For example, if DS1 has element A as a char, the DS1 Java class is as follows:

```
Public class DS1
{
    public void setA()
    {
        ...
    }
    public char getA()
    {
        ...
    }
}
```

The data structure can contain two kinds of compound objects, JDEDate and JDEMathnumeric, in addition to the primitive data types. The two Java classes JDEDate and JDEMathnumeric are defined respectively.

Note

If this is your first implementation of a Java connector, PeopleSoft suggests you consider the dynamic Java connector instead of the Java connector. The functionality is the same. The advantage of implementing the dynamic Java connector is that you are not required to generate wrappers.

JDEDate

JDEDate()	Construct a JDEDate
getDay()	Get the day of the date
getMonth()	Get the month of the date
getYear()	Get the year of the date
setDay(short)	Set the day of the date
setMonth(short)	Set the month of the date
setYear (short)	Set the year of the date

JDEMathNumeric

getValue()	Return the value as a String (for example, "-12345.6789")
setValue(String strValue)	Set the value from a String (for example, "-12345.6789")
getCurrencyDecimals()	Get the Currency Decimal positions
setCurrencyDecimals(int aValue)	Set the Currency Decimal positions
getCurrencyCode()	Get the Currency Code
setCurrencyCode(String aValue)	Set the Currency Code
getDecimalPosition()	Get the Decimal Position
isNegative()	Test if the value is negative
reset()	Reset all the internal values

To set the value of a member in a MathNumeric type in a data structure, use the method `setValue(String)` in `JDEMathNumeric` class. For example, if `mnAddressBook` is a member in the data structure, then a class should exist for the data structure with the public method `getmnAddressBook`, which returns a `JDEMathNumeric` object. Then you use `DS.getmnAddressBook().setValue("1")` to set the `mnAddressBook` value to 1 in the data structure.

Design Time for the Java Connector

This section covers considerations for designing your Java connector solution.

GenJava

PeopleSoft provides a Java generation tool, GenJava, that you can run to expose business functions through Java. A system administrator usually runs GenJava.

When you run GenJava, you specify a library of business functions, for example CAEC, to wrap. GenJava creates Java class files for all the business functions and associated data structures. GenJava also compiles the business functions, generates Java docs, and packages them to two JAR files, one for Java classes and one for Java documents. For example, if the library is `JDEAddressBook`, you see `JDEAddressBookInterop.jar` and `JDEAddressBookInteropDoc.jar` in either the `B9\system\classes` directory or any directory redirected by GenJava.

Understanding Java Versioning

Business object wrappers that are generated against one environment might not be compatible with another environment. Versioning prevents you from creating Java business objects unless the environment used at logon is the same as the environment used to generate the wrappers, or the environment is compatible with the business objects. You can use the Java Wrapper Version Checker (CheckVer) to verify that business object wrappers are compatible with new environments.

Migrating from Previous Releases

Previously generated business object wrappers are compatible with the new versioning code; you do not need to regenerate them. However, in order to use them, CheckVer must be run, even for the environment used to create the wrappers. The repository setting in the [INTEROP] section of the ini file must point to the directory containing the jar files of generated business object wrappers. For example:

```
[INTEROP]
repository=c:\foo\bar\repository
```

The repository directory should contain only jar files for generated business object libraries.

Java Connector Static and Dynamic Modes

A Java interoperability client can be configured statically or dynamically. Static mode is the normal mode of operation and should be used by most client code. Dynamic mode is better suited for developing tools based on Java interoperability. The two modes can be used simultaneously in the same process. The granularity is at the business object library (jar file) level. No matter which mode is used, it is necessary for the jar files to be placed in the repository directory.

To use a business object library in static mode, ensure that the jar file is in the classpath and in the repository directory for the client process.

To use dynamic mode for a given business object library, ensure that the jar file is in the repository directory but not in the classpath. Dynamic mode is for Java interoperability clients with client code that has no direct use of the business objects. In dynamic mode, business objects may only be used via the classes in the java.lang.reflect package. However, dynamic mode allows client code to refresh, add, or remove business object libraries while in operation. These operations are accomplished using the methods in the OneWorldVersion class. For example, generate a new business object library (or regenerate an existing library) using GenJava. Use the CheckVer tool to establish the compatible environments for the business objects in the library. Add the jar file to the repository directory. Finally, the client code must instantiate a OneWorldVersion object, and call the refreshLibrary() method. To remove a business object library, remove it from the repository and call the refreshLibrary() method.

After a library is refreshed, all newly created business objects use the new definition. Business objects created before the refresh use the old definition. No limit exists for the number of simultaneous business object library versions. The old library definitions remain in the virtual machine until no more references to the old business objects exist, which can significantly affect memory use in the virtual machine.

Using the Java Wrapper Version Checker (CheckVer)

You can run CheckVer to verify whether a previously generated Java business object library is compatible with another environment. Typically, the system administrator performs this task. The XML files generated by GenJava are the signatures of the objects generated against specific PeopleSoft EnterpriseOne environments. These XML files can be used with CheckVer to verify that the wrappers in a previously generated jar file are compatible with these environments.

When you introduce a new PeopleSoft EnterpriseOne environment, you run GenJava against the new environment by using the /XMLOnly option. You also use the iJDEScript that you used to generate the wrappers to generate XML signature files for the objects in the new environment. Run CheckVer with the new XML files and previously generated jar files to verify that the new environment is compatible with the wrappers. CheckVer updates the jar file according to the result of the compatibility test. A Java client using the jar file can be dynamically updated to the new compatibility information, using the OneWorldVersion interface. If the new environment is incompatible, the client is not allowed to create business objects with the new environment.

Running CheckVer (GenJava)

CheckVer is a Java class and should not be confused with the CheckVer.exe that is a part of the COM interoperability solution. CheckVer takes two arguments, the jar file name and the XML file name. CheckVer requires that the Connector.jar, Kernel.jar, xalan.jar, and xerces.jar files be in the CLASSPATH. This can be done either with the CLASSPATH environment variable or on the command line itself.

Syntax

Java com.jdedwards.system.connector.CheckVer [*jarfile*] [*xmlfile*]

Example

Java com.jdedwards.system.connector.CheckVer *JDEAddressBookInterop.jar* *JDEAddressBook.xml*

Setting Up a Client Environment for GenJava

You must take several steps to set up a client environment for GenJava. You should make sure the PATH environment variable and the CLASSPATH environment variable are set up correctly.

PATH

<bin directory for JDK>

Example: c:\jdk1.2.2\bin

CLASSPATH

<Directory where PeopleSoft EnterpriseOne is located>\System\classes\Kernel.jar

<Directory where PeopleSoft EnterpriseOne is located>\System\classes\Connector.jar

<Directory where PeopleSoft EnterpriseOne is located>\System\classes\Xalan.jar

<Directory where PeopleSoft EnterpriseOne is located>\System\classes\Xerces.jar

Installing a Java Connector

The following steps show how to install Java connector components so that you can run a Java connector application on a non-PeopleSoft EnterpriseOne machine.

1. Copy the following files from the PeopleSoft EnterpriseOne server to a directory on the desired machine. For example, copy the following files to c:\PeopleSoft\Interop on your machine:
 - kernel.jar
 - connector.jar
 - jdeinterop.ini
 - xalan.jar
 - xerces.jar
 - database.jar
 - Log4j.jar
 - jdelog.properties
 - JDBC driver (you need to get JDBC driver from the vendor)
2. Add the following files to the CLASSPATH:
 - kernel.jar
 - jdeutil.jar
 - connector.jar
 - database.jar
 - log4j.jar
 - xerces.jar
 - JDBC driver
3. Add the path where the jdelog.properties and jdeinterop.ini files are located into CLASSPATH.
4. Create a separate repository directory for business object.jar files.
5. Run GenJava on the client machine and copy the output jar file (for example, JDEAddressBook.jar) to this directory.
6. Depending on whether you want the library in static mode or dynamic mode, put the business object.jar file in the CLASSPATH.

Runtime for the Java Connector

This section covers runtime considerations for your Java connector.

Using the Java Generator (GenJava)

The Java generation tool, GenJava, provides access to business functions by generating Java interfaces for business functions. GenJava includes the following components:

- GenJava.exe
- Emitter framework
- JDEIDJavaEmitter.dll

You use iJDEScript scripting language to script code generation activities when you use GenJava.

Running GenJava

You run GenJava from the command line. There are several options available for generation. GenJava is located in <install>\system\bin32.

Syntax

GenJava [options] [libraries]

Options

/?	Lists the options available for generation.
/Cat <category>	Generates only <category> function wrappers. Supports the following categories: /1/ - Master Business Functions /2/ - Major Business Functions /3/ - Minor Business Functions /-/ - Uncategorized Business Functions
/Cmd *	Processes code generation commands from the console.
/Cmd <filename>	Processes code generation commands from <filename>.
/Compiler <file>	Uses <file> to compile Java files.
/D name value	Defines a macro value.
/EnvironmentID <env>	Uses <env> to sign in to PeopleSoft EnterpriseOne.
/ListLibraries	Lists the available libraries that you can use for GenJava.

/MsgFile <file>	Provides GenJava with the filename to log messages produced by GenJava during the generation process; for example, "messages.log."
/NoBSFN	Tells GenJava not to create wrappers for business functions. This option is for generating parameter sets only.
/Out <path>	Provides GenJava with the directory (path) in which to place the output files; for example, "C:\winnt\system32."
/Password <password>	Provides GenJava with the password with which you want to sign in to PeopleSoft EnterpriseOne.
/Role	Provides GenJava with the role with which you want to sign in to PeopleSoft EnterpriseOne.
/TempOut <path>	Provides GenJava with the directory (path) in which to place temporary files needed for the build process; for example, "C:\temp."
/UserID <userid>	Provides GenJava with the username with which you wish to sign in to PeopleSoft EnterpriseOne.
/XMLOnly	Generates only the XML file.

You can also use GenJava by running it with a JDEScript file, such as:

```
GenJava /cmd AddressBook.cmd
```

This command prompts a sign-in window for you to enter the user ID, password, role, and environment. The AddressBook.cmd is as follows:

```
define library JDEAddressBook

login

library JDEAddressBook

interface AddressBook

import B0100031

import B0100019

import B0100032

import B0100002
```

```
import B0100033
```

```
build
```

```
logout
```

GenJava generates the wrappers in Java for all business functions imported in the script file.

Generate Java Wrappers

The following command generates Java wrappers for Category 1 business functions in the CAEC library:

```
GenJava /Cat 1 /UserID Devuser1 /Password Devuser1 /Environment ADEVHP02 CAEC
```

You must use the correct information (including user ID, password, role, and environment) to log on to PeopleSoft EnterpriseOne.

Using GenJava Output

The output for GenJava produces fully functional Java objects based on the library you use to generate wrappers. GenJava packages these objects in a single jar file such as XXXXInterop.jar or XXXXInteropDoc.jar, where XXXX is the library name defined in the script file or from the command line. For example, JDEAddressBookInterop.jar is created for the above AddressBook.cmd. The default location for the jar file is under B9/System/classes, but it can be somewhere else if you run GenJava using a /Out value. This jar file must be deployed to the machine that uses those wrappers. To import any wrapper object/class, the jar file must be added to the CLASSPATH. Because you are interacting with the PeopleSoft EnterpriseOne system, three components, Connector.jar, Kernel.jar, and jdeinterop.ini file, must be deployed to the machine.

XXXXInteropDoc.jar is the compressed format of all the Java documents (html files) for all the classes generated by GenJava.unjar. You can also unzip the jar file to see the APIs that can be called in these classes.

All Java client applications must do the following:

1. Initialize a com.jdedwards.system.connector.Connector.
2. Sign in to PeopleSoft EnterpriseOne using a valid user ID, password, role, and environment name. The environment must be valid on the PeopleSoft EnterpriseOne server.
3. Get the OneWorldInterface object reference by calling Connector.CreateBusinessObject() with an object name, such as Connector::OneWorldInterface.
4. Get the object reference for the wrapper for the business function generated by GenJava, for example AddressBook. The object name passed into Connector.CreateBusinessObject should be "Library (Java package) Name:Object Name," such as "JDEAddressBook:AddressBook."
5. Call CreateXXXParameterSet() on the wrapper object for any data structure XXX.
6. Set the needed value in the data structure.

7. Call the business function with the data structure variable as a parameter. Check the return value. The return value can be one of the following:

Successful = 0

Warning = 1

Error = 2

Process the data returned by the business function.

8. Disconnect from PeopleSoft EnterpriseOne.

The following examples illustrate how to use a generated Java business function wrapper in a Java application.

```
import com.jdedwards.system.connector.*;
import com.jdedwards.application.interop.jdeaddressbook.*;
public class abclient
{
    public static void main (String[] args) {
        Connector connectorProxy = null;
        OneWorldInterface ow;
        AddressBook ab;
        D0100033 ds;
        sessionID=0;
1.      connectorProxy = new Connector();
        try {
2.          sessionID = connectorProxy.Login("FOO", "BAR", "PDEVHPO2");
            System.out.println("Log in successfully");
        } catch (reject r) {
            System.out.println("got reject exception");
            String s = r.reason;
            System.out.println(s);
            System.exit(1);
        } catch (Exception e) {
            System.out.println("got other exception");
            e.printStackTrace();
            System.exit(1);
        }
        try {
3.          ow = (OneWorldInterface)connectorProxy.CreateBusiness Object
                ("Connector:: OneWorldInterface", sessionID)
            System.out.println("got OneWorldInterface");
        } catch (reject r){
            String s = r.reason;
            System.out.println(s);
            return;
        }
        //create AddressBook object
        try {
4.          ab = (AddressBook)connectorProxy.CreateBusinessObject
                ("JDEAddressBook:: AddressBook", sessionID)
            System.out.println("got AddressBook");
        } catch (reject r) {
```

```

        String s = r.reason;
        System.out.println(s);
        return;
    }
    // get data structure D0100033
5.    ds = ab.CreateGetEffectiveAddressParameterSet();
    // set addressbook number value in D0100033
6.    ds.getmnAddressNumber().setValue("1")
    // get address information
    int i = 0;
    try {
7.        i = ab.GetEffectiveAddress(ds, ow, connectorProxy; sessionID);
    } catch (reject e) { System.out.println(e.reason); }
    if (i!=2){
        String alphaname = ds.getszNamealpha();
        String address = ds.getszAddressLine1();
        String zipcode = ds.getszZipCodePostal();
        String city = ds.getszCity();
        String county = ds.getszCountyAddress();
        String state = ds.getszState();
        String country = ds.getszCountry();
        If (i==1){
            System.out.println("warning count is"
                               +ow.GetWarningCount());
            for ( int j = 0; j<ow.GetWarningCount(); j++) {
                String s = ow.GetWarningAt(j);
                System.out.println("warning" + j + ";" + s)
            }
        }
    } else {
        for (int j = 0; j<ow.GetErrorCount(); j++){
            String s = ow.GetErrorAt(j);
            System.out.println("error" + j + ";" + s);
        }
        System.out.println("BSFN error");

        //log off
8.        connectorProxy.Logoff(1);
    } // end main
} // end abclient

```

Transaction via Java Connector

Transactions are a way to update the PeopleSoft EnterpriseOne database. You can use the Java connector to do a transaction in either auto mode or manual mode. When you use auto transaction mode, the transaction is immediately committed after the business function call is completed. The transaction is set to the auto commit mode by default. When you use manual transaction mode, the transaction is started by explicitly calling `BeginTransaction` in `OWInterface`, and the transaction is committed (or rolled back) by calling `Commit`(or `Rollback`) in `OWInterface`.

Note

The PeopleSoft EnterpriseOne transaction is not really a two-phase commit. You need to manually roll back the transaction when the commit statement is reached.

The following example shows a basic manual commit transaction:

```
import com.jdedwards.system.connector.*;

public class ConnectorDemo {
    public static void main(String argv[]) {
        OWInterface ow;
        try {
            Connector con = new Connector();
            int accessnumber = con.login("User", "Password", "Env", "Role");
            ow = (OneWorldInterface)
con.CreateBusinessObject("Connector::OneWorldInterface", 1);
            // ... handle the message
            ow.BeginTransaction(con, accessNumber);
            soe.F4211FSBeginDoc(soeBeginDoc,ow, con, accessnumber);
            soe.F4211FSEditLineDoc(soeEditLine,ow, con, accessnumber);
            soe.F4211FSEditLineDoc(soeEditLine,ow, con, accessnumber);
            soe.F4211FSEndDoc(soeEndDoc,ow, con, accessnumber);
            ow.Commit();
        }catch (Exception e) {
            ow.rollback();
        }
    }
}
```

Using BHVRCOM via Java Connector

You can use the BHVRCOM structure to control the execution of business functions. You use the Java connector to call methods in the OWInterface class to set and pass the BHVRCOM fields to business functions on the server. The following table shows the business function methods and the BHVRCOM fields:

Business Function Method	BHVRCOM Field
setBOBMode(int bobMode)	IBobMode
setAPPName(String aName)	szApplication
setUserName(String aName)	szUser
setDatabaseChanged(Boolean value)	bDataBaseChange

The following Java code demonstrates how to query the IBHVRCOM interface and pass values to business functions:

```
...
    ow = (OneWorldInterface)
connectorProxy.CreateBusinessObject("Connector::OneWorld Interface", 1);
ab= (AddressBook) connectorProxy.CreateBusinessObject("JDEAddress Book::Address
Book", 1);
ds.getmnAddressNumber().setValue("1");
ow.setAppName("AddressbookApp");
ow.setBOBMode(8);
ow.setUserName("Java Connector");
ow.SetDatabaseChanged(false);
    i = ab.GetEffectiveAddress(ds, ow, connectorProxy, 1);
...
```

OCM Support for the Java Connector

You use Object Configuration Manager (OCM) to map business functions to a PeopleSoft EnterpriseOne server so that the Java connector can access OCM to run business functions. You no longer configure the jdeinterop.ini file to define the PeopleSoft EnterpriseOne server from which you want to execute business functions. Using OCM support should result in an increase in performance, scalability, and load balancing. The Java interoperability server distributes the processes of the Java client to various PeopleSoft EnterpriseOne servers depending on user, environment, and role. To take advantage of Java connector OCM support, you must do the following:

- Use a B9 or later version of GenJava to regenerate the business wrapper function.
- Configure your OCM and map your business function on different PeopleSoft EnterpriseOne servers.
- Set OCMEEnabled=true in jdeinterop.ini.
- Configure the settings in jdeinterop.ini regarding the bootstrap data source with your OCM configuration.

Ensure that the following settings in your jdeinterop.ini configuration file are set:

jdeinterop.ini File Section	Required Settings
OCM	OCMEEnabled
JDBj-BOOTSTRAP SESSION	user, password, environment, and role
JDBj-BOOTSTRAP DATA SOURCE	name, databaseType, server, database, serverPort, physicalDatabase, library, owner
[JDBj-JDBC DRIVERS]	ORACLE, iSeries, SQLSERVER, UDB
[JDBj-ORACLE]	tns

The dynamic Java connector and the Java connector use OCM in the same way.

User Session Management for the Java Connector

When the connector user successfully signs on, a valid user session is allocated to that user signon. The user session has status for two types of connector operations: one for inbound business function calls and the other for outbound real-time events. The connector monitors the status of the user session, and uses the timeout settings in the jdeinterop.ini file to stop the user session when a timeout setting has been reached. The connector looks at the following settings:

jdeinterop.ini File Section	Setting	Explanation
[CACHE]	UserSession	The maximum connector idle time for an inbound business function call.
[INTEROP]	manual_timeout	The maximum idle time for a manual transaction.
[EVENTS]	outbound_timeout	The maximum value of connector idle time for receiving outbound events.

The value for the settings is in milliseconds. A value of zero (0) indicates infinite timeout. The above settings are defined in the jdeinterop.ini section of this guide.

If an inbound user session times out, that user session cannot be used to execute a business function call. Likewise, if an outbound user session times out, that user session cannot be used for events. When both inbound and outbound sessions time out, the user session is removed from the connector. Since each user session has a corresponding handle in the PeopleSoft EnterpriseOne server, PeopleSoft *highly* recommends that you explicitly call a Connector API to log off the user session to release the handle in the PeopleSoft EnterpriseOne server when the user session is no longer used.

The following sample codes shows how to retrieve and manage a user session:

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

// Do whatever using the sessionID. If InvalidSessionException is caught, user
session is not valid any more

//Check the status of the usersession
UserSession session;
try{
    session=Connector.getInstance().getUserSession(sessionID);
}catch(InvalidSessionException ex){
    System.out.println("Invalid user session");
}
if(session.isInboundTimedout()){
    System.out.println("User session inbound is timed out");
}
if(session.isOutboundTimedout()){
    System.out.println("User session outbound is timed out");
}

//Log off and shut down connector to release user session from the server
connector.logoff(sessionID);
connector.shutdown();
```

The dynamic Java connector and the Java connector handle XML requests in the same way.

Inbound XML Request via Java Connector

You can use the Java connector to send inbound synchronous XML requests (such as XML CallObject and XML List) to the PeopleSoft EnterpriseOne server. The Java connector has an API that it calls to send XML documents to JDENET.

The following example code shows how to use the Java connector to execute an inbound XML request:

```
Connector conn = new Connector();
//...login into OW
String xmlDoc;
//or byte[] xmlDoc
//...Load a String or byte[] into xmlDoc;

String requestResult = conn.executeXMLRequest(xmlDoc);
//...handle requestResult.
```

The Dynamic Java connector and the Java connector handle XML requests in the same way.

See Also

See the following topics in the *Interoperability Guide*:

- ❑ *XML CallObject* for more information about using XML CallObject
- ❑ *XML List* for more information about using XML List
- ❑ *XML Transaction* for more information about using XML Transaction

Exception Handling for the Java Connector

When you run the Java connector or the GenJava tool, the program might encounter a condition that causes unexpected results or system failure. When the program does not perform as expected, an error occurs; or, using Java terminology, an exception is thrown. In Java, the system, classes, and programs can throw exceptions. You can write code to catch exceptions. Catching an exception involves dealing with the exception conditions so that your program will not crash.

All exceptions in the connector and GenJava code inherit from the reject class. Your program needs to catch only the reject exception conditions for the methods that throw exceptions. PeopleSoft created the FatalException class and the RecoverableException class so that you can provide a recovery action in your program for some exceptions, thereby minimizing manual intervention.

Fatal Exception

FatalException class conditions are unlikely or impossible to resolve without manual intervention. If you catch fatal exception conditions in your program, you can include a string message that indicates the condition that occurred. You use the getMessage() method from the java.lang.Throwable class to retrieve fatal exception messages from your program. The system uses the INTEROP category to log fatal exception conditions to the jas.log file.

Recoverable Exception

You can provide the capability for the system to possibly resolve an exception condition by catching RecoverableException (and children) class conditions in your program. The children of recoverable exception conditions indicate through their class names the category of the exception and include a string message in the constructor to provide more exception details. You use the getMessage() method from the java.lang.Throwable class to retrieve recoverable exception messages from your program. The system uses the INTEROP category to log recoverable exception conditions to the jasdebug.log file. You can turn off recoverable exception messages through the DEBUG flag in the jdeinterop.ini file. The flag is either true or false.

Reject

The method signature for each of the methods listed in the following table indicates that the method only throws “reject,” even though the exceptions thrown in each method’s code are children of the reject class. Even if you decide to catch all of the exceptions listed in Exception Details table which follows, you will also need to catch “reject” as the last in the series of connector-related catch statements because of the method signatures’ stated throws clause.

Exception Details

The methods that throw exceptions in each of the main public classes of the connector (Connector, OneWorldInterface, EventSource, and GenJava-created business object code) are detailed in the following table. The information in this table is also available in the Javadoc for the connector, which is in the ConnectorDoc.jar file.

Class	Method	Exception	Condition	Possible Action
Connector	Login	CallObjectRetryException	The error code returned by CallObject is TIMEOUT or RETRY_NEEDED	Retry Login method
		CallObjectIgnoreException	The error code returned by CallObject is NOERROR, ALREADY_EXECUTED, or BAD_ERRORPACKETS	Ignore this exception
		FatalException	The error code returned by CallObject is any other error code	*
	CreateBusinessObject	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
		FatalException	A Java reflection exception is thrown or the PeopleSoft EnterpriseOne environment is not in sync with the business function wrapper	*
OneWorldInterface	GetNextError	NoMoreDataException	Error index reaches the end of the array	End the loop searching for the next error
	GetNextWarning	NoMoreDataException	Warning index reaches the end of the array	End the loop searching for the next warning
	Commit	InvalidMethodCallException	This method is called before PrepareToCommit() is called	Call the PrepareToCommit() method
		CallObjectRetryException	The error code returned by CallObject is TIMEOUT or	Retry Commit method

Class	Method	Exception	Condition	Possible Action
			RETRY_NEEDED	
		CallObjectIgnoreException	The error code returned by CallObject is NOERROR, ALREADY_EXECUTED , or BAD_ERRORPACKETS	Ignore this exception
		FatalException	The error code returned by CallObject is any other error code	*
	Rollback	CallObjectRetryException	The error code returned by CallObject is TIMEOUT or RETRY_NEEDED	Retry Rollback method
		CallObjectIgnoreException	The error code returned by CallObject is NOERROR, ALREADY_EXECUTED , or BAD_ERRORPACKETS	Ignore this exception
		FatalException	The error code returned by CallObject is any other error code	*
	PrepareToCommit	CallObjectRetryException	The error code returned by CallObject is TIMEOUT or RETRY_NEEDED	Retry PrepareToCommit method
		CallObjectIgnoreException	The error code returned by CallObject is NOERROR, ALREADY_EXECUTED , or BAD_ERRORPACKETS	Ignore this exception
		FatalException	The error code returned by CallObject is any other error code	*
	ExecuteBSFN	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
		CallObjectRetryException	The error code returned by CallObject is TIMEOUT or RETRY_NEEDED	Retry ExecuteBSFN method

Class	Method	Exception	Condition	Possible Action
			RETRY_NEEDED	
		CallObjectIgnoreException	The error code returned by CallObject is NOERROR, ALREADY_EXECUTED, or BAD_ERRORPACKETS	Ignore this exception
		FatalException	The error code returned by CallObject is any other error code	*
EventSource	EventSource (Constructor)	FatalException	The connector cannot listen on the given port	*
	addListener	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
		FatalException	The subscription fails	*
	removeListener	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
		FatalException	The unsubscription fails	*
	updateSession	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
	getEventTemplate	NotLoggedInException	The user is not currently logged in to PeopleSoft EnterpriseOne	Log in through Connector class
		FatalException	A JdeNetException is thrown	*
	getEventTypes	NotLoggedInException	The user is not currently logged on to PeopleSoft EnterpriseOne	Log in through Connector class
		FatalException	A JdeNetException is thrown	*
GenJava-created Data Structures	setString<parameter> methods	StringTooLongException	The value set for the parameter is too long	Reset the parameter using a shorter length

For `FatalException` conditions, you can send the exception message, which can be retried by using the `getMessage()` method, to your system administrator. Alternatively, you can prompt your system administrator to look in the `jas.log` file for more details about the exception. It is unlikely that your program can recover associated PeopleSoft EnterpriseOne or connector errors during runtime.

Example: Java Connector Exception Handling Sample Code

The following code illustrates some of the features of the enhanced connector exception handling. The bold-faced items indicate specific exception-handling code.

```
import com.jdedwards.system.connector.*;
import com.jdedwards.application.interop.jdeaddressbook.*;

public class AddressClient {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Must supply a city to query for AddressBook");
            System.exit(-1);
        }

        Connector connectorProxy = null;
        OneWorldInterface ow = null;
        AddressBook ab = null;
        D0100033 ds = null;

        int accessNumber = 0;
        connectorProxy = new Connector();

        try {
            accessNumber = connectorProxy.Login("FOO", "BAR", "PDEVHP02");
            System.out.println("Logged in successfully");
        } catch (CallObjectIgnoreException e) {
            // do nothing
        } catch (CallObjectRetryException e) {
            // try one more time
            try {
                accessNumber = connectorProxy.Login("FOO", "BAR", "PDEVHP02");
                System.out.println("Logged in successfully");
            } catch (CallObjectIgnoreException ex) {
                // do nothing
            } catch (CallObjectRetryException ex) {
                System.out.println("EXCEPTION: " + ex.toString());
                System.out.println("Nested Exception: " +
ex.getChainedException().toString());
                System.out.println("Refer to the jasdebug.log file for more details.");
                System.exit(-1);
            } catch (FatalException ex) {
                System.out.println("Fatal Exception during login: " + ex.toString());
                System.out.println("Refer to the jas.log file for more details.");
                System.exit(-1);
            } catch (reject r) {
                System.out.println("Java Connector Exception: " + r.reason);
                System.exit(-1);
            }
        } catch (FatalException e) {
```

```

        System.out.println("Fatal Exception during login: " + e.toString());
        System.out.println("Refer to the jas.log file for more details.");
        System.exit(-1);
    } catch (reject r) {
        /* This should not happen, as the Java Connector code
         * now only throws one of the reject child objects.
         * The documentation indicates which methods throw which
         * reject child exception objects. All methods continue
         * to have a signature of "throws reject", however, for
         * backwards compatibility (to not break existing client code).
         */
        System.out.println("Java Connector Exception: " + r.reason);
        System.exit(-1);
    }

    try {
        ow =
        (OneWorldInterface)connectorProxy.CreateBusinessObject("Connector::OneWorldInterface",
        accessNumber);
        System.out.println("Got OneWorldInterface");
    } catch (FatalException e) {
        System.out.println("Fatal Exception during OneWorldInterface creation: " +
        e.toString());
        System.out.println("Refer to the jas.log file for more details.");
        System.exit(-1);
    } catch (reject r) {
        System.out.println("Java Connector Exception: " + r.reason);
        System.exit(-1);
    }

    try {
        ab =
        (AddressBook)connectorProxy.CreateBusinessObject("JDEAddressBook::AddressBook",
        accessNumber);
        System.out.println("Got AddressBook");
    } catch (FatalException e) {
        System.out.println("Fatal Exception during OneWorldInterface creation: " +
        e.toString());
        System.out.println("Refer to the jas.log file for more details.");
        System.exit(-1);
    } catch (reject r) {
        System.out.println("Java Connector Exception: " + r.reason);
        System.exit(-1);
    }

    ds = ab.CreateGetEffectiveAddressParameterSet();

    ds.getmnAddressNumber().setValue("1");

    try {
        ds.setszCity(args[0]);
    } catch (StringTooLongException e) {
        System.out.println("Cannot set a city with length of " + args[0].length());
        System.exit(-1);
    }

```

```

    } catch (reject r) {
        System.out.println("Java Connector Exception: " + r.reason);
        System.exit(-1);
    }

    int i=0;

    try {
        i = ab.GetEffectiveAddress(ds, ow, connectorProxy, accessNumber);
    } catch (CallObjectIgnoreException e) {
        // do nothing
    } catch (CallObjectRetryException e) {
        // try one more time
        try {
            i = ab.GetEffectiveAddress(ds, ow, connectorProxy, accessNumber);
        } catch (CallObjectIgnoreException ex) {
            // do nothing
        } catch (CallObjectRetryException ex) {
            // don't try again after second try
            System.out.println("EXCEPTION: " + ex.toString());
            System.out.println("Nested Exception: " +
ex.getChainedException().toString());
            System.out.println("Refer to the jasdebug.log file for more details.");
            System.exit(-1);
        } catch (FatalException ex) {
            System.out.println("Fatal Exception during AddressBook retrieval: " +
ex.toString());
            System.out.println("Refer to the jas.log file for more details.");
            System.exit(-1);
        } catch (reject r) {
            System.out.println("Java Connector Exception: " + r.reason);
            System.exit(-1);
        }
    } catch (FatalException e) {
        System.out.println("Fatal Exception during AddressBook retrieval: " +
e.toString());
        System.out.println("Refer to the jas.log file for more details.");
        System.exit(-1);
    } catch (reject r) {
        System.out.println("Java Connector Exception: " + r.reason);
        System.exit(-1);
    }

    String alphaname = ds.getszNamealpha();
    String address    = ds.getszAddressLine1();
    // get whatever other AddressBook parameters you desire...

    if (i == 1) { // business function warning
        System.out.println("Warning count is " + ow.GetWarningCount());
        for (int j=0; j<ow.GetWarningCount(); j++) {
            System.out.println("Warning " + j + ": " + ow.GetWarningAt(j));
        }
    } else if (i == 2) { // business function error
        for (int j=0; j<ow.GetErrorCount(); j++) {
            System.out.println("Error " + j + ": " + ow.GetErrorAt(j));
        }
    }

```

```

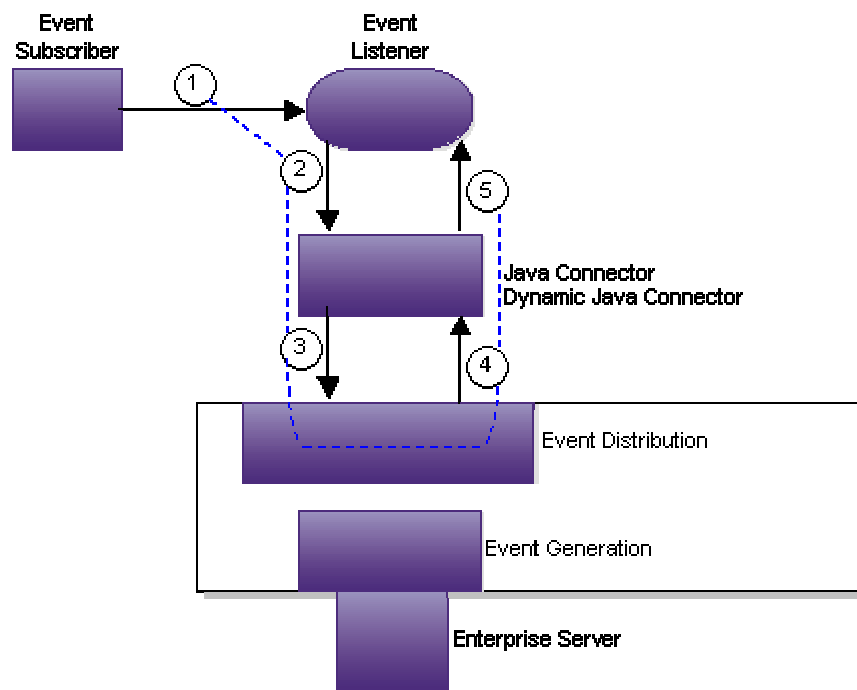
    }
    connectorProxy.Logoff (accessNumber) ;
}
}

```

Java Connector Outbound Events

The Java connector outbound event source architecture enables Java clients to use either the Java connector or the dynamic Java connector to subscribe to various transaction types in PeopleSoft EnterpriseOne and receive notification upon completion of those transactions. For example, a client can subscribe to the event JDESOOUT and then receive notification when a sales order transaction is complete in PeopleSoft EnterpriseOne.

The following diagram illustrates this process:



1. PeopleSoft EnterpriseOne clients create different types of EventListeners.
2. PeopleSoft EnterpriseOne clients subscribe to various event types with the Java connector.
3. When the Java connector receives a subscription for a given event, it subscribes to the same event type with the event distribution kernel.
4. PeopleSoft EnterpriseOne events originate from the real-time events kernel or from callback functions in Uses. When the event distribution kernel receives an event to which the Java connector has subscribed, it sends the event to the Java connector.
5. The Java connector sends the event to all subscribers for that event. The EventListener callback function is executed to receive the subscribed event.

Note

The outbound events architecture is the same for the Java connector and the dynamic Java connector. The difference is that corresponding package location for the dynamic Java connector is `com.jdedwards.system.connector.dynamic.*` and `com.jdedwards.system.connector.dynamic.events.*`.

For purposes of discussion in this document, the Java connector is used to illustrate the outbound events architecture. Special notes are added to discuss any API differences between the Java connector and dynamic Java connector.

Do not mix the usage of APIs from the two connectors in one application.

Developing the Java Client to Use the Java Connector Outbound Event Source

You can use the Java connector outbound event source to subscribe to an outbound event. The following list identifies the tasks for setting up and using your Java client to subscribe to PeopleSoft EnterpriseOne transaction types and notify you upon completion of the transaction.

- Create a Java class to implement an interface.
- Create a Java client application to subscribe to an event.
- Compile and run the Java client.

Implement an Interface

You create a Java class to implement an interface to PeopleSoft EnterpriseOne. Depending on the purpose for which you are using the Java class, implement one of the following interfaces:

- `com.jdedwards.system.connector.events.CountedListener`
Implement this interface if you want to know the subscription count, when the subscription count is reached, and when the subscribed event is dropped.
- `com.jdedwards.system.connector.events.PersistentListener`
Implement this interface if you want the real-time event kernel to persist the subscription when the kernel goes down and comes up, and the connection to the Java connector is re-established.
- `com.jdedwards.system.connector.events.EventListener`
Implement this interface for most other situations.

No matter which interface you implement, the implementation Java class must contain the following five methods:

```
//set the event type to subscribe
void setEventType( String type );
String getEventType();

// stop/start the event coming in the Java connector
void setPause( boolean pause );
boolean isPaused();

// the callback function when the event arrives
void onOneWorldEvent( EventObject event );
```

Subscribe to an Event

You create a Java Client Application to subscribe to an event. Your Java Client Application must do the following:

1. Create a new instance of the Connector class.
2. Use the Connector object to verify the client's user ID, password, and environment, and then log the client into PeopleSoft EnterpriseOne.
3. Do one of the following:
 - Java Connector: Create an EventSource object by calling the CreateBusinessObject method of the Connector class, passing in an "Events::EventSource" string identifier.
 - Dynamic Java Connector: Get EventSource instance by calling `com.jdedwards.system.connector.dynamic.connector.events.EventSource.getInstance()`.
4. Create an EventListener object.
5. Specify the specific event type to which to subscribe.
6. Register the EventListener object with the EventSrc object.
7. Develop a callback function. When the subscribed to event arrives, the EventListener calls this callback function. This step is optional.

Example: Using the Java Client to Subscribe to an Event Using the Java Connector Outbound Event Source

The following example illustrates how to write code for a Java client to subscribe to an event using the Java connector outbound event source.

```
import java.io.*;
import javax.swing.*;
import com.jdedwards.system.connector.*;
import com.jdedwards.system.connector.events.EventListener;

/**
 * The event source client application
 */
class EventClient
{
    private Connector m_connector = null;
    private int m_Access = 0;
    private EventSource m_theSource = null;
    private Listener m_listener = null;

    public static void main(String argv[]) {
        try
        {
            // 1.
            m_connector = new Connector();

            // 2.
            m_Access = m_connector.Login("user", "password",
"environment");

            // 3.
            // passing in an "Events::EventSource" string identifier.
            m_theSource = (EventSource)m_connector.CreateBusiness
Object("Events:: EventSource", m_Access);

            // 4.
            m_listener = new ListenerImpl(this);

            // 5.
            m_listener.setType("JDESOUT");

            // 6.
            m_theSource.addListener( m_listener, m_Access );
        }
        catch(Exception e)
        {
            System.out.println(e.toString())
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```

// 7.
public synchronized void executeCallBack(EventObject event){
    System.out.println("Getting the event:"+event.getData());
    //execute the call back function;
}

}

/**
 * The EventListener interface is the means by which events are
 * delivered to the client by the Java connector.
 * The client have to implement an EventListener object
 */

public class ListenerImpl implements EventListener
{
    String m_eventType;
    boolean m_paused = false;
    EventClient m_client;

    /** Creates new Listener */
    public ListenerImpl()
    {
    }
    public ListenerImpl(EventClient client)
    {
        this.m_client=client;
    }

    public synchronized String getEventType()
    {
        return m_eventType;
    }

    public void setEventType(java.lang.String eventType)
    {
        this.m_eventType = eventType;
    }

    public synchronized boolean isPaused()
    {
        return m_paused;
    }
    public synchronized void setPause(boolean pause)
    {
        m_paused = pause;
    }

    public synchronized void onOneWorldEvent(EventObject p1)
    {
        System.out.println("Received event: " + p1.getType());
        // if the arrival event is the one that client subscribes,
        // the EventListner can trigger the call back function in the client
        if (p1.getType().equalsIgnoreCase(m_eventType)) {
            m_client.executeCallBack(p1);
        }
    }
}

```

```
}  
}  
}
```

Compile and Run the Java Client

To compile the Java client, use the following command:

```
set JAVA_HOME = <the path of JDK>  
set OneWorld_HOME = <the installation path>  
set CLASSPATH=%OneWorld_HOME%\system\classes\Kernel.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\Connector.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_Home%\system\classes\log4j.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\xalan.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\xerces.jar  
%JAVA_HOME%\bin\javac -classpath %CLASSPATH% EventClient.java  
EventListenerImpl
```

To run the Java client, use the following command:

```
set JAVA_HOME = <the path of JDK>  
set OneWorld_HOME = <the installation path>  
set CLASSPATH=%OneWorld_HOME%\system\classes\Kernel.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\Connector.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_Home%\system\classes\log4j.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\xalan.jar  
set CLASSPATH=%CLASSPATH%;%OneWorld_HOME%\system\classes\xerces.jar  
%JAVA_HOME%\bin\java -classpath %cp%  
EventClient
```

J2EE Connector Architecture Resource Adapter

The PeopleSoft EnterpriseOne J2EE Connector Architecture (JCA) resource adapter allows Java2 Platform, Enterprise Edition (J2EE) components to use a standard interface to connect to the PeopleSoft EnterpriseOne system. A resource adapter is a system-level software driver that enables J2EE components to communicate with a back-end enterprise information system (EIS) through a JCA-compliant application server when a resource adapter for the specific EIS is deployed to the server. J2EE components consist of Servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs).

J2EE components and applications built with J2EE components can execute business functions through the PeopleSoft EnterpriseOne JCA resource adapter. PeopleSoft EnterpriseOne business functions are accessed through the JCA standard client interface, the Common Client Interface (CCI). The PeopleSoft EnterpriseOne JCA resource adapter is fully compliant to the Java2 Platform, Enterprise Edition (J2EE) JCA 1.0 Specification and should work with any application server that is J2EE 1.3 certified.

Note

Some application servers are known to not be J2EE 1.3 certified, but they support some J2EE 1.3 features, including JCA 1.0. Check with your application server vendor to determine whether your application server supports JCA 1.0.

See Also

- ❑ *J2EE Connector Architecture* (<http://java.sun.com/j2ee/connector/>) for more information about J2EE Connector architecture

JCA 1.0 Specification Optional Features

The JCA 1.0 Specification identifies optional features for developing a resource adapter. The following table addresses the level of support that the PeopleSoft EnterpriseOne JCA resource adapter provides for the optional features identified in the JCA 1.0 Specification.

Transactions	The PeopleSoft EnterpriseOne JCA resource adapter is classified as an XA Transaction resource adapter. The PeopleSoft EnterpriseOne JCA resource adapter allows either no transactions during business function calls, transactions local to PeopleSoft EnterpriseOne during those same calls (local transaction), and one-phase commit (1PC) XA transactions (transactions that span multiple enterprise information systems).
Client Interface	The PeopleSoft EnterpriseOne JCA resource adapter supports the optional common client interface (CCI), which is modeled after the Java Database Connectivity (JDBC) client API. This relatively simple Java API should significantly reduce the learning curve for using the PeopleSoft EnterpriseOne JCA resource adapter.
Reauthentication	The PeopleSoft EnterpriseOne JCA resource adapter does not support the switching of a set of PeopleSoft EnterpriseOne user credentials on an existing PeopleSoft EnterpriseOne user session. User credentials are usually a concern of the application server and should not affect client development.
Input/Output Records	The PeopleSoft EnterpriseOne JCA resource adapter supports the MappedRecord interface, which is a data type of key-value pairs. The MappedRecord interface is further discussed in the Input/Output Data section of this document. The CCI interfaces IndexedRecord and ResultSet are not supported as they are not relevant to the type of output from business functions.

Authentication	The PeopleSoft EnterpriseOne JCA resource adapter supports BasicPassword authentication, which indicates to the application server how to handle container-managed signon. The resource adapter does not support any other form of authentication, such as Kerberos authentication through the GenericCredential interface. The Signon Types section of this document provides more information about authentication with the resource adapter.
ManagedConnectionFactory Properties	The JCA Specification identifies the following properties as standard; however, these properties are optional properties for the ManagedConnectionFactory class, which is the main class configured with PeopleSoft EnterpriseOne specific properties during deployment of the resource adapter: • ServerName • PortNumber • UserName • Password • ConnectionURL The PeopleSoft EnterpriseOne JCA resource adapter supports the UserName and Password properties, as the other properties are either irrelevant properties or are configured elsewhere in the resource adapter. The Deployment Settings section of this document addresses other properties that are defined by the PeopleSoft EnterpriseOne JCA resource adapter. Note The deployment tool of your particular J2EE application server might list the above properties as configurable for the resource adapter. The PeopleSoft EnterpriseOne JCA resource adapter will not use values that you assign to these properties (other than that for UserName and Password).
Number of Deployed Resource Adapters	The JCA Specification allows for the possibility of deploying the same resource adapter multiple times on a given application server. This provides for potential connectivity to multiple versions of the same EIS for a one resource adapter-to-many-EIS version ratio. The PeopleSoft EnterpriseOne JCA resource adapter supports the deployment of only one PeopleSoft EnterpriseOne JCA resource adapter per application server (essentially one resource adapter per virtual machine). Note You can install different JCA resource adapters (those other than for PeopleSoft EnterpriseOne) on the same application server.

Non-Managed Scenario	The PeopleSoft EnterpriseOne JCA resource adapter must be used with an application server or an application client. If you want to access PeopleSoft EnterpriseOne business functions through Java outside of an application server or application client, you should use the Java connector directly.
-----------------------------	--

See Also

- ❑ The online *JCA Javadoc* (http://java.sun.com/j2ee/apidocs-1_0-fr/api/index.html) for official online JCA Java documentation (APIs)

Assembly and Components

The packaging of a resource adapter is defined in the JCA 1.0 Specification. However, because some application servers require additions to the standard Resource Adapter Archive (RAR) file, it is not possible to distribute a single RAR file that can be deployed to all application servers. The components that you need to assemble for the PeopleSoft EnterpriseOne JCA resource adapter are identified below. Consult your application server documentation for instructions on how to use the assembly tool and to understand what additional components might be required for a resource adapter to be operational with your application server. Typically, an additional deployment descriptor is required. Additional information required by an application server is usually for performance tuning and for configuration settings.

Components

A RAR file is a file that is in Java Archive (JAR) File Format with a .rar extension instead of a .jar extension. The file structure for a RAR file is:

- /META-INF/ra.xml
- /<all necessary JAR files>

The ra.xml file is the standard resource adapter deployment descriptor and must be put in the META-INF directory of the RAR file. The ra.xml file must be named *exactly* ra.xml. The ra.xml file for the PeopleSoft EnterpriseOne JCA resource adapter is provided in the system/classes/samples directory on the PeopleSoft EnterpriseOne CD. All other JAR files go in the root directory of the RAR file. The JAR files are provided in the system/classes directory on the same CD. The required resource adapter JAR files include:

- owra.jar
- Connector.jar
- database.jar
- jdeutil.jar
- kernel.jar
- log4j.jar
- xerces.jar
- JDBC driver .jar files supplied by your database vendor

Note

Only use the versions of these JAR files that come with your PeopleSoft EnterpriseOne distribution.

When your RAR file is finally created, the META-INF directory of your RAR file might contain a Manifest.mf file. The Java JAR tool usually creates the Manifest.mf file automatically. The Manifest.mf file complies with the JAR file format, and it is acceptable for the Manifest.mf file to be in the RAR file.

Deployment and Configuration

The methods and tools for configuring and deploying a resource adapter vary between application servers and even between versions of the same application server. Consult your application server's documentation for information about how to configure and deploy a resource adapter. Two separate methods exist for deploying a resource adapter. The first method is deploying the resource adapter as a standalone resource adapter. This allows all applications deployed on the application server to access the same resource adapter. The second method involves packaging the resource adapter within an enterprise application (EAR) file. This allows only those components in the EAR file to have access to the resource adapter. The sample applications provided with the PeopleSoft EnterpriseOne JCA resource adapter use the second method.

Additional settings required for the PeopleSoft EnterpriseOne JCA resource adapter to be deployed and to operate correctly include the following:

- Security permissions
- jdeinterop.ini settings
- jdbj.ini settings
- jdelog.properties settings
- CLASSPATH settings
- Configurable properties
- JNDI settings

Note

Only one PeopleSoft EnterpriseOne JCA resource adapter can be deployed in standalone mode per application server.

Security Permissions

The JCA 1.0 Specification defines the standard Java security permissions that must be granted to all resource adapters by an application server. The PeopleSoft EnterpriseOne JCA resource adapter needs additional security permissions in order to operate. These permissions are listed in the deployment descriptor (ra.xml file). Most application servers dynamically grant these permissions to the resource adapter during deployment. Some application servers have other methods of granting the resource adapter additional permissions, including modifying a Java security policy file, which might require that you restart your application server to take effect.

If your application server does not dynamically grant the security permissions to a resource adapter based on the contents of the deployment descriptor, you will need to grant the resource adapter the permissions listed in the security-permission-spec elements of the deployment descriptor. If your application server throws a `SecurityException` while running an application associated with the PeopleSoft EnterpriseOne JCA Resource Adapter, it is possible that the necessary security permissions are not being granted to the resource adapter.

jdeinterop.ini Settings

Because the resource adapter is built on top of the Java connector, it is necessary to configure the appropriate settings in your `jdeinterop.ini` file to make the Java connector operational. The resource adapter introduces no new settings into the `jdeinterop.ini` file.

jdbj.ini Settings

You must set up the `jdbj.ini` file.

See Also

- ❑ *Appendix D, Parameters and Values for the jdbj.ini file and Sample JDBj.ini file* in the *Web Server Installation Guide* for information on how to setup the `jdbj.ini` file

jdelog.properties Settings

The JCA 1.0 Specification allows for resource adapter-specific logging messages to be sent to a separate log file, which can be configured according to your application server (see your application server documentation). The messages that are sent to this log file will be redundant to and are a subset of the messages that are sent to the log file defined in the `jdelog.properties` file. This redundancy is an intentional PeopleSoft EnterpriseOne JCA resource adapter design decision for the following reason:

The JCA logging mechanism does not provide a method for logging messages from the connector on which the resource adapter is built. The logging properties file allows all logging messages from the connector as well as the resource adapter to be logged in a central location.

See Also

- ❑ *jdelog.properties* in the *Connectors Guide* for information on how to establish the proper settings for this file

CLASSPATH Settings

The PeopleSoft EnterpriseOne JCA resource adapter requires that the complete path to your `jdelog.properties` file be placed in your server's CLASSPATH. This path cannot include the name of the file, and the path must end with a slash, which designates that the last item in the path is a directory and not a file. The name of the properties file is required to be *jdelog.properties*. The logging mechanism will look for the logging properties file in all directories in the CLASSPATH.

The JDBC driver for your PeopleSoft EnterpriseOne database needs to be in your server's CLASSPATH so that the proper database connections can be made.

Note

Some servers require all of the JAR files within the resource adapter RAR file to be placed in the server's CLASSPATH. If you encounter a *NoClassDefFoundError* while running a Web application that is using the resource adapter, try putting all of these JAR files in the server's CLASSPATH and restarting the server. Consult your server documentation for further ClassLoader issues.

Configurable Properties

The PeopleSoft EnterpriseOne JCA resource adapter deployment descriptor (ra.xml file) contains properties that must be assigned values specific to your environment. The following table identifies the configurable properties and describes the information required.

owVersion	The version of PeopleSoft EnterpriseOne to which the resource adapter will connect, such as PeopleSoft EnterpriseOne 8.93. This property is for display purposes only and can contain any value. The value you enter in this property is not validated against your PeopleSoft EnterpriseOne installation.
username	Use this property for a PeopleSoft EnterpriseOne user when neither the container nor the application supplies a set of PeopleSoft EnterpriseOne user credentials.
password	Use this property for a PeopleSoft EnterpriseOne user when neither the container nor the application supplies a set of PeopleSoft EnterpriseOne user credentials.
environment	It is possible in a resource adapter web application to map a user's web credentials to a set of PeopleSoft EnterpriseOne user credentials. This mapping, which is called container-managed sign-on, prevents the user from having to present different credentials multiple times while using a single web application. Container-managed sign-on maps a given username and password to a PeopleSoft EnterpriseOne username and password. Container-managed signon mapping is specific to each application server. For PeopleSoft EnterpriseOne, the environment property is used to add a valid PeopleSoft EnterpriseOne environment to the username and password mapped by the application server, which allows for proper PeopleSoft EnterpriseOne signon. If you use container-managed signon, you must assign a value to this property.
role	In addition to username, password, and environment, PeopleSoft EnterpriseOne signon requires a role. The role property has a default value of <i>*ALL</i> , which allows the user to assume all valid roles for the PeopleSoft EnterpriseOne username. You do not need to assign a value for role if this is the value you want to use.

Consult the documentation for your application server to determine if other deployment settings are required.

Java Naming and Directory Interface Settings

For communication between your web application and the PeopleSoft EnterpriseOne JCA resource adapter, the web application must perform a Java Naming and Directory Interface (JNDI) lookup of the ConnectionFactory of the resource adapter. You are allowed to configure multiple ConnectionFactory instances for each resource adapter. This allows for setting different values for the configurable properties listed in the previous section. The web application obtains a PeopleSoft EnterpriseOne connection and interacts with PeopleSoft EnterpriseOne through the ConnectionFactory. The method of assigning a JNDI name to the ConnectionFactory for the PeopleSoft EnterpriseOne JCA resource adapter is specific to and documented by your application server.

When you add a ConnectionFactory through your application server, you are provided with a method for assigning values to the configurable properties for each ConnectionFactory.

Common Client Interface

The Common Client Interface (CCI) is the JCA-recommended client API for all resource adapters. The PeopleSoft EnterpriseOne JCA resource adapter provides an implementation of CCI as the client interface. The following code example, along with subsequent paragraphs, show how to use the CCI API.

Example: Common Client Interface Code Sample

The following example code shows how to implement a CCI for the PeopleSoft EnterpriseOne JCA resource adapter. In the example, the elements in quotes have descriptive names and must have values valid to your environment in a real Java class. The line numbers in the example code are not part of the code but are for reference in subsequent paragraphs.

```
import com.jdedwards.system.connector..dynamic.jcaplugin.ImageBSFNInteractionSpecImpl;
import com.jdedwards.system.jca.cci.ConnectionSpecImpl;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.resource.ResourceException;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Interaction;
import javax.resource.cci.MappedRecord;
import javax.resource.cci.RecordFactory;
import javax.resource.cci.ResourceWarning;

public class SomeClass {

    public void someMethod() {

        try {
            // get the naming context
            Context nc = new InitialContext();

            // lookup the connection factory
            ConnectionFactory conFact = (ConnectionFactory)nc.lookup("Resource Adapter JNDI Name");
```

```

        // create a ConnectionSpec
1.    ConnectionSpecImpl conSpec = new ConnectionSpecImpl("username", "password",
"environment", "role");

        // get the Connection to PeopleSoft EnterpriseOne
2.    Connection con = conFact.getConnection(conSpec);

        // create an Interaction
Interaction ix = con.createInteraction();

        // create and populate the InteractionSpec
OWBSFNInteractionSpecImpl ixSpec = new OWBSFNInteractionSpecImpl();
ixSpec.setBusinessFunction("Business Function Name");

        // get a RecordFactory
RecordFactory rf = conFact.getRecordFactory();

        // create the input MappedRecord
3.    MappedRecord inputRecord = rf.createMappedRecord("any descriptive name");

        // populate the input MappedRecord with the input values
4.    inputRecord.put("Business Function Parameter Name",
        " Business Function Parameter Value");

        // execute the Business Function, putting the results in the output
// MappedRecord
5.    MappedRecord outputRecord = (MappedRecord)ix.execute(ixSpec, inputRecord);

        // get results
Object value = outputRecord.get("Business Function Parameter Name");

        // get Business Function warnings, if any
ResourceWarning warning = ix.getWarnings();

        // close the Interaction
ix.close();

        // close the Connection
con.close();
    } catch (ResourceException e) {
        // handle resource adapter-related Exceptions here
    } catch (NamingException e) {
        // handle JNDI-related Exceptions here
    }
}
}

```

Signon Types

The PeopleSoft EnterpriseOne JCA resource adapter provides the following types of PeopleSoft EnterpriseOne signons:

- Container-managed signon
- Component-managed signon

Container-managed signon

When container-managed signon is used, the application server maps a web application user to a given PeopleSoft EnterpriseOne user. In this case, PeopleSoft EnterpriseOne user credentials are not provided in the CCI code. If you use container-managed signon, line 1 of the example code would not exist, as you do not need to create an instance of the `ConnectionSpecImpl` class. Line 2 of the example code would be changed to the following:

```
Connection con = conFact.getConnection();
```

Component-managed signon

When component-managed signon is used, the code provides specific PeopleSoft EnterpriseOne credentials (either through coding specific credentials or by obtaining PeopleSoft EnterpriseOne credentials through user entry in the web application) to the PeopleSoft EnterpriseOne JCA resource adapter for PeopleSoft EnterpriseOne signon. In the example code, lines 1 and 2 illustrate component-managed signon. As shown in line 1 of the example code, an instance of the `ConnectionSpecImpl` class is first created with the PeopleSoft EnterpriseOne user credentials. That instance is then passed to the `getConnection` method, which is shown in line 2.

Component-managed signon is also known as application-managed signon.

See Also

- ❑ *Configurable Properties* in the PeopleSoft EnterpriseOne JCA Resource Adapter section of the *Connectors Guide* for more information about container-managed signon

Subclasses

The import statements at the top of the example code show that most of the classes that you use to interact with the PeopleSoft EnterpriseOne JCA resource adapter are JCA classes (those classes in the `javax.resource` package and sub-packages) and not PeopleSoft-specific implementations of JCA interfaces. PeopleSoft provides the following implementation classes:

- `ConnectionSpecImpl`
- `xxxxInteractionSpecImpl`

The `ConnectionSpecImpl` class supplies the required PeopleSoft EnterpriseOne user credentials to the `getConnection` method. The `ConnectionSpecImpl` class is one of the signon types. Line 1 in the example code shows how to use the `ConnectionSpecImpl` class.

The purpose of the `xxxxInteractionSpecImpl` class is to establish the necessary business function information before execution in PeopleSoft EnterpriseOne. The `xxxxInteractionSpecImpl` classes vary, depending on the type of business function spec source. The business function spec source is a file or location that describes a business function. Each implementation class, which is a concrete class of the `javax.resource.cci.InteractionSpec` interface, includes methods that set values. These “setter” methods must be called and given valid values before executing the business function through the resource adapter.

ImageBSFNInteractionSpecImpl

The `ImageBSFNInteractionSpecImpl` implementation class gets the business function spec from an XML image file, which must be generated by the dynamic Java connector beforehand.

Class: `com.jdedwards.system.connector.dynamic.jcapugin.ImageBSFNInteractionSpecImpl`

Method: `setBusinessFunction(String value)`

Sets the *exact* name of the business function.

Method: `setImageFilename(String value)`

Sets the complete path and filename of the dynamic Java connector PeopleSoft EnterpriseOne spec image that contains the definition of the corresponding business function.

OWBSFNInteractionSpecImpl

The `OWBSFNInteractionSpecImpl` implementation class gets the business function spec directly from a call to PeopleSoft EnterpriseOne. This method might take a little longer to execute a business function the first time the business function is called. The business function is stored in memory, and execution should be quicker in subsequent calls.

Class: `com.jdedwards.system.connector.dynamic.jcapugin.OWBSFNInteractionSpecImpl`

Method: `setBusinessFunction(String value)`

Sets the *exact* name of the business function.

Input and Output Data

The `MappedRecordImpl` class handles both sending input data to the resource adapter and receiving the output data that is the result of executing the business function. Lines 3 and 4 of the example code illustrate inputting data, and line 5 illustrates obtaining the output data. A `MappedRecord` is a correlation of key/value pairs. The key represents the exact business function parameter name, and value defines the key.

Input data for values can be supplied in one of the following ways:

- Use a `String`
- Use a native Java data type

The PeopleSoft EnterpriseOne JCA resource adapter examines the input data on a parameter-by-parameter basis. If the input data type is `String`, the resource adapter will attempt to convert the input data to the appropriate Java data type for the specified parameter. If both the actual parameter type and the input data are `String`, the resource adapter will pass the input data through unchanged. If the input parameter is a native Java data type, the resource adapter will pass the input data through unchanged.

If the native Java data type is incorrect or if the parameter name is invalid for the given business function, the resource adapter will throw an Exception.

The following table lists the business function types and their corresponding native Java data type:

PeopleSoft EnterpriseOne Data Type	Native Java Data Type
ID	java.lang.Integer
char (length of only 1)	java.lang.Character
JDEDATE	java.util.Date
Calendar	com.jdedwards.base.datatypes.JDECalendar (located in Kernel.jar file)
MATH_NUMERIC	com.jdedwards.system.lib.MathNumericImpl (located in Kernel.jar file)
char (variable length)	java.lang.String

The output of all business functions will result in the data in the MappedRecordImpl being in the native Java data types. If you prefer deal with only String-formatted output, you can make the following call on the output MappedRecordImpl for each parameter retrieved:

```
String value = outputRecord.get("parameter name").toString();
```

Logging

Message logging for the PeopleSoft EnterpriseOne JCA resource adapter is controlled by the jdelog.properties file, which was discussed previously.

Exceptions

The parent Exception class for all exceptions thrown by the PeopleSoft EnterpriseOne JCA resource adapter is javax.resource.ResourceException.

See Also

- ❑ *JCA Javadoc* (http://java.sun.com/j2ee/apidocs-1_0-fr/api/index.html) for the complete set of Exception subclasses and the methods that throw each Exception

Samples

The samples supplied with the resource adapter show how to use the resource adapter's API, as well as the JCA API, in general and how to demonstrate the functionality of the resource adapter. Address Book Query, Sales Order Entry, and Purchase Order Entry are included samples. The source code along with the compiled classes are delivered on your PeopleSoft EnterpriseOne Java Server CD in the system/classes/samples directory in the JCASamples.ear file and the JCASamples_WebSphere.ear file, which includes WebSphere 5.x-specific deployment files.

The sample applications consist of a group of servlets, which provide the HTML for the display of the samples, and a group of stateful session Enterprise JavaBeans (EJBs) that access the PeopleSoft EnterpriseOne JCA resource adapter. The resource adapter is bundled inside the .ear files, and is only available to the sample applications when deployed to your application server.

Prepare the Samples for Deployment

Several customizations must be performed to the .ear file before it will function correctly.

JDBC Driver .jar File

The JDBC driver .jar file supplied by your PeopleSoft EnterpriseOne database vendor must be packaged inside the .ear file. Since the .ear file is in a Zip format, you can use a Zip program to add the necessary files. Place the JDBC driver .jar files in the root directory of the .ear file (no path for those files). The CLASSPATH in the manifest.mf file on the .ear file includes the expected filenames for the JDBC .jar files for three database vendors without actually being included in the driver files themselves:

- SQL Server: msbase.jar, msutil.jar, mssqlserver.jar
- Oracle: classes12.jar
- DB2: jt400.jar

If the file names of your JDBC driver .jar files are different, add those file names to the manifest.mf file that is located inside the meta-inf directory of the JCA Samples RAR.rar file within the sample application.ear file you are using. Be sure to preserve the meta-inf path for the manifest.mf file when you add it back into the file.

Configuration Files

You must configure the following files using the settings that are provided in this document:

- jdbj.ini
- jdeinterop.ini
- jdelog.properties.

These configuration files are in the config directory of the sample application EAR file. After you customize the settings, be sure to place the files back into the EAR file in the config directory.

Customize the Samples for the Application Server

A generic JCASamples.ear file and a WebSphere 5.x-specific JCASamples_websphere.ear file are provided. Your application server might need additional information for some of the components contained in the EAR file. The following is a list of the sample components:

- JCASamplesEJB.jar
A JAR file that contains the Enterprise JavaBean (EJB) classes used by the samples.
- JCASamplesRAR.rar
A rar file that contains only the resource adapter deployment descriptor. The dependent JAR files for the resource adapter are contained in the parent directory of the EAR file, as they need to be used by the entire application.

- JCSamplesWeb.war

A WAR file containing the servlets for the sample applications.

If you use the generic JCSamples.ear file to deploy the sample applications to your application server, and they do not operate correctly, you might need to unpack each of the files individually (.ear, .jar, .rar, and .war files) and repack them with your application server's assembly tool. This step usually allows the tool to place new files and information in existing files that allow the application to operate correctly for that application server.

Deploy the Sample Applications

The following general steps must be completed for deploying the sample applications to any application server. Your application server may require additional steps not listed here (see your application server's documentation for deploying enterprise applications):

- Start your application server.
- Start your administrative console (whatever application that ships with your application server that allows you to deploy applications).
- Install the enterprise application.
- Add a connection factory for the resource adapter with a JNDI name of OneWorldJCAAdapter (with that exact spelling).
- Restart your application server.

Deploy the Sample Applications to WebSphere 5.x

Use the following steps to deploy the sample applications on WebSphere 5.x.

1. Start WebSphere.
2. Start the WebSphere Administrative Console.
3. Log on to the WebSphere Application Server Administrative Console using any ID.
4. On the WebSphere Administrative Console, expand the applications node on the left side of the screen, and then click the Install New Applications link.

Preparing for the Application Installation appears on the right side of the screen.

5. In the Preparing for the application installation portion of the screen, click Browse, and then choose JCSamples_WebSphere.ear file.
6. Click Next.

Continue to click Next on all successive screens until the final Summary screen presents a Finish button at the bottom of the screen. Accept the defaults provided on each of the screens without altering any of them.

7. On the final Summary screen, click Finish.

WebSphere automatically generates the necessary EJB deployment code.

8. At the bottom of the screen, after the notice that the Application JCA samples installed successfully, click the Save to Master Configuration link.
A Message box (indicating that changes have been made to your local configuration) and an Enterprise Application Save section that includes a Save to Master Configuration box appear.
9. In the Save to Master Configuration box, click the Save button.
You are returned to the main screen.
10. On the left side of the main screen, click the Enterprise Applications link from the menu.
A list of the installed applications appears on the right side of the screen.
11. On the right side of the screen, click JCASamples from the list that appears under Enterprise Applications.
12. In the Related Items area at the bottom of the next screen, click the Connector Modules link.
13. On Connector Modules, click JCASamplesRAR.rar.
14. On the screen with a Configuration tab, scroll to the Additional Properties area and click the Resource Adapter link.
15. On the next screen with a Configuration tab, scroll to the Additional Properties area, and then click the J2C Connection Factories link.
16. On the J2C Connection Factories screen, click the New button to establish a new J2C Connection Factory.
17. Under the Configuration tab on the New screen, enter any value for the Name field and OneWorldJCAAdapter for the JNDI name.
The value you enter for the Name field is used for display purposes only.
18. Scroll to the bottom of the screen, and then click the OK button.
19. In the Message box at the top of J2C Connection Factories screen, click the Save link.
20. In the Save to Master Configuration area on the Save screen, click the Save button.
21. From the menu bar (at the top of the screen), click Logout.
22. Stop and restart your sever to make the application is available to run.

Run the Sample Applications

After you configure and deploy the sample applications, you can run each of the sample applications, provided that the PeopleSoft EnterpriseOne Server you are accessing is operational. Use the following URLs to access the samples:

- AddressBook Query: `http://<app http://<app server name>|<app server port>/JCASamplesWeb/ABLogin`
- SalesOrder Entry: `http://<app server name>|<app server port>/JCASamples Web/SOLogin`
- PurchaseOrder Entry: `http://<app server name>|<app server port>/JCASamplesWeb/POLogin`

Troubleshooting Checklist

- The directory location of your `jdelog.properties` file must be in your server's CLASSPATH. For example, if your `jdelog.properties` file is in the following location:

`C:\JCA\logs\jdelog.properties`

you must have the following entry in your server's CLASSPATH:

`C:/JCA/logs/`

Be sure to include a slash at the end of the path to indicate that *logs* is a directory and not a file. When you make a change to your server's CLASSPATH, you must restart your server.

- Some servers read the `<security-permission-spec>` element of the resource adapter's deployment descriptor (the `ra.xml` file) and dynamically grant the resource adapter the security permissions listed in those elements. If you are executing a resource adapter-based application and experience a `java.xxx.xxxPermission Exception`, you will have to manually add the contents of the `<security-permission-spec>` elements to your server's policy file. Consult your server's documentation for the location and format for editing the policy file. You should be able to simply copy and paste the elements into the server's policy file. Any changes to the policy generally require a server restart to take effect.

If you make the changes and still experience Permission Exceptions, you might need to move some of the permission elements that you copied from the *resource adapter* domain in the policy file to the *default domain* in the policy file. This is because the resource adapter classes, especially if present in the server's CLASSPATH, might reside in the *default domain* and not the *resource adapter* domain.

jdeinterop.ini

The jdeinterop.ini file includes settings the server might need. The default location for the file is `c:\`; however, this location is configurable. This section details the settings found in the jdeinterop.ini file. Information is organized by section, for example [JDENET].

[OCM]

The following settings are used only by the COM connector.

Setting	Typical Value	Purpose
DSN=	ODA ITTND17	The data source name from the system DSN of your ODBC setting.
OCM Datasource=	COM OCM	System data source for PeopleSoft EnterpriseOne client.
DB User=-	jde	User for the data source connection.
DB Pwd=	jde	Password for the data source connection.
Object Owner=	sysb9	For UNIX platforms, this is the object owner in the [DB SYSTEM SETTINGS].
Seperator=	.	Separator used in SQL query. For Oracle, SQL, and UDB databases, the separator is period (.); for iSeries, the separator is /.

The following setting is used only by the dynamic Java connector.

Setting	Typical Value	Purpose
OCMEnabled=	True	Turn on or turn off OCM inside the Java connector. A value of <i>true</i> indicates turned on.

[CACHE]

The following settings are used only by the Java interoperability connectors.

Setting	Typical Value	Purpose
UserSession=	0	Timeout value (in milliseconds) for the dynamic Java connector user session. A zero (0) indicates infinite timeout.
SpecExpire=	30000000	Maximum time (in milliseconds) that dynamic Java connector will keep the fetched spec in the cache.

[JDENET]

Setting	Typical Value	Purpose
enterpriseServerTimeout=	90000	Timeout value for a request to the PeopleSoft EnterpriseOne server.
maxPoolSize=	30	JDENET socket connection pool size.
serviceNameConnect=	6004	Port number used by the PeopleSoft EnterpriseOne security server. This setting is used only by the Java connector.

[SERVER]

Setting	Typical Value	Purpose
glossaryTextServer=	JDED:6010	The PeopleSoft EnterpriseOne server and port that provide glossary text information.
codePage=	1252	The encoding scheme, such as: 1252 English and Western European 932 Japanese 950 Traditional Chinese 936 Simplified Chinese 949 Korean

[SECURITY]

Setting	Typical Value	Purpose
NumServers=	1	Number of security servers set.
SecurityServer=	JDED	The PeopleSoft EnterpriseOne security server. This setting is used only by the Java connector.

[DEBUG]

The [DEBUG] section is used only by the COM connector.

Setting	Typical Value	Purpose
JobFile=	c:\Interop.log	Location of error file
DebugFile=	c:\InteropDebug.log	Location of debug file
log=	c:\net.log	Location of log file
debugLevel=	0 - 12	Defines the level of tracing provided by the COM connector and the Callobject component in the specified log file, in the COM server only. 0 None. Logging is turned off and only errors are written to the JobFile. 2 Errors (error messages). 4 System Errors (exception messages). 6 Warning Information. 8 Min Trace (Key operations; for example, Login, Logoff, Business Function calls.) 10 Trouble Shooting Information (Help). 12 Complete Debug Information (Logs everything). Note: - The odd values are reserved for future levels to be added. - You typically do not need tracing on by default. However, tracing is useful for debugging.

Setting	Typical Value	Purpose
netTraceLevel=	0	Defines the level of tracing provided by the ThinNet component in the specified log file, in the COM server only. 0 No trace. 1 Record process ID, thread ID, and the available socket status when a new connection is added and the socket pool is searched. 2 Includes the information in trace level 1 and also traces every call made in the connection manager class. 3 Includes all information in trace level 2, and also traces getPort() calls and getHost() calls. Note: You typically do not need tracing on by default. However, tracing is useful for debugging.

[INTEROP]

Setting	Typical Value	Purpose
enterpriseServer=	JDED	The PeopleSoft EnterpriseOne server.
port=	6010	The port number of the PeopleSoft EnterpriseOne server.
manual_timeout=	300000	The timeout value for a transaction in manual commit mode.
Repository	c:\PeopleSoft\Interop\repository	Points to the location of the repository directory containing business object libraries (generated JAR files).

[EVENTS]

Setting	Typical Value	Purpose
port=	6002	The socket port number where the EventListener will receive the events from the PeopleSoft EnterpriseOne server. This port should not be used by any other resource. Also, the port should not be changed dynamically when the connector is running, as this will cause subsequent subscriptions to be lost.
ListenerMaxConnection=	10	The maximum number of connections allowed by the EventListener. The default number of connections is 10, but you can change this number. The maximum number of connections allowed is 64.
ListenerMaxQueueEntry=	10	The maximum number of events that the EventListener can hold before processing by the EventManager. The default number of events for the queue is 10, but you can change this number. The maximum number of events that can be held in the queue is 100.
Outbound_timeout	1200000	Maximum number of milliseconds that the EventManager waits before unsubscribing the transient event from the PeopleSoft EnterpriseOne server.

[JDBj-BOOTSTRAP SESSION]

The JDBj sections are used only by the Java interoperability connectors. The JDBj sections define the OCM bootstrap data source for accessing the OCM bootstrap tables. These sections also define database connection configuration needed by JDBj (PeopleSoft EnterpriseOne Database implementation). Settings could vary depending on the type of database.

The [JDBj-BOOTSTRAP SESSION] section defines the bootstrap session. Bootstrap session information is used for signon to provide access to system tables. JDBj logs you on through the security server at bootstrap time. If the session information is not valid, you will not be able to access the PeopleSoft EnterpriseOne database.

Setting	Typical Value	Purpose
User=	JDE	User name for database access.
Password=	JDE	Password for database access.
Environment=	PDEV	The PeopleSoft EnterpriseOne environment.
Role=	*ALL	The PeopleSoft EnterpriseOne user role.

[JDBj-BOOTSTRAP DATA SOURCE]

The JDBj sections are used by the Java interoperability connectors only. The JDBj sections define the OCM bootstrap data source for accessing the OCM bootstrap tables. These sections also define database connection configuration needed by JDBj (PeopleSoft EnterpriseOne Database implementation). Settings could vary depending on the type of database.

The [JDBj-BOOTSTRAP DATA SOURCE] section defines the physical bootstrap data source information. Bootstrap data source information defines the data source where the OCM resides. JDBj uses this at bootstrap time and later to look up OCM entries on demand. If the data source information is not valid, you will not be able to access the PeopleSoft EnterpriseOne database.

Note

The information in the [JDBj-BOOTSTRAP DATA SOURCE] section should match the corresponding information specified in the [DB SYSTEM SETTINGS] section of your installation's JAS.INI and JDE.INI files.

Not all of the settings are required for all platforms. The following table provides an explanation of the settings.

Setting	Typical Value	Purpose
name=	System – B9	Data source name
server=	JDE	Database server name
DatabaseType=		Database type. DatabaseType depends on the platform: - iSeries - databaseType=I - Oracle – databaseType=O - SQL Server – databaseType=S - UDB – databaseType=W
database=		Database
serverPort=		Database server port
physicalDatabase=		Physical database name
owner=	SystemB9	Database owner

Not all settings are needed for all platforms. The following table identifies the required settings by platform:

Platform	Required Settings
iSeries	name, databaseType, server, physicalDatabase
Oracle	name, databaseType, database, owner
SQL Server	name, databaseType, server, serverPort, physicalDatabase, owner
UDB	name, databaseType, database, owner

[JDBj-CONNECTION POOL]

The JDBj sections are used only by the Java interoperability connectors. The JDBj sections define the OCM bootstrap data source for accessing the OCM bootstrap tables. These sections also define database connection configuration needed by JDBj (PeopleSoft EnterpriseOne Database implementation). Settings could vary depending on the type of database.

The [JDBj-CONNECTION POOL] section defines the connection pool settings. These settings are used only when J2EE connection pooling is not available.

Setting	Typical Value	Purpose
jdbcPooling=	False	Allows JDBC pooling
minConnection=	5	Minimum physical connection in the connection pool
maxConnection=	5	Maximum physical connection in the connection pool
initialConnection=	5	Initial connection when started up
poolGrowth=	5	The connection size to increase when pool size grows
connectionTimeout=	1800000	The timeout value, in milliseconds, for a connection to be made
cleanPoolInterval=	600000	Interval time when connection pool is cleaned

[JDBj-JDBC DRIVERS]

The JDBj sections are used only by the Java interoperability connectors. The JDBj sections define the OCM bootstrap data source for accessing the OCM bootstrap tables. These sections also define database connection configuration needed by JDBj (PeopleSoft EnterpriseOne Database implementation). Settings could vary depending on the type of database.

The [JDBj-JDBC DRIVERS] section defines the JDBC drivers to be used by the Java interoperability connectors. You only need to set the setting that is applicable for your database type.

Setting	Typical Value	Purpose
ORACLE=	Oracle.jdbc.driver.OracleDriver	JDBC driver for Oracle
iSeries=	Com.ibm.as400.access.AS400JDBCDriver	JDBC driver for iSeries
SQLSERVER=	Com.Microsoft.jdbc.sqlserver.SQLServerDriver	JDBC driver for MS SQL Server
UDB=	COM.ibm.DB2.jdbc.app.DB2Driver	JDBC driver for UDB

[JDBj-ORACLE]

The JDBj sections are used by the Java interoperability connectors only. The JDBj sections define the OCM bootstrap data source for accessing the OCM bootstrap tables. These sections also define database connection configuration needed by JDBj (PeopleSoft EnterpriseOne Database implementation). Settings could vary depending on the type of database.

The [JDBj-ORACLE] section defines Oracle tnsnames location. You only need to set this setting when your database type is Oracle.

Setting	Typical Value	Purpose
tns=	C:\Oracle\Ora901\network\ADMIN\tnames.ora	Oracle tnsnames location

jdelog.properties

The logging utility in the dynamic Java connector, the Java connector, and Java connector Architecture (JCA) is built on top of Apache open source project Log4j. The jdelog.properties file defines the settings for the logging configuration. The jdelog.properties file should be physically located in CLASSPATH.

The following settings provide a sample configuration file for root configuration:

- Root configuration

```
jdelog.rootLogger=DEBUG,JDELOG,JASLOG
```

```
jdelog.loggerFactory=com.jdedwards.base.logging.log4j.JdeLoggerFactory
```

```
jdelog.reloadInterval=60
```

- File handler for root log

```
jdelog.handler.JDELOG=com.jdedwards.base.logging.log4j.FileHandler
```

```
jdelog.handler.JDELOG.File=\\jderoot.log
```

```
jdelog.handler.JDELOG.Level=ERROR
```

```
jdelog.handler.JDELOG.Append=TRUE
```

```
jdelog.handler.JDELOG.MaxBackupIndex=1
```

```
jdelog.handler.JDELOG.MaxFileSize=10MB
```

```
jdelog.handler.JDELOG.format=com.jdedwards.base.logging.log4j.DefaultFormat
```

- File handler setting for jas log

```
jdelog.handler.JASLOG=com.jdedwards.base.logging.log4j.FileHandler
```

```
jdelog.handler.JASLOG.File=\\jas.log
```

```
jdelog.handler.JASLOG.Level=ERROR
```

```
jdelog.handler.JASLOG.Append=TRUE
```

```
jdelog.handler.JASLOG.MaxBackupIndex=1
```

```
jdelog.handler.JASLOG.MaxFileSize=10MB
```

```
jdelog.handler.JASLOG.format=com.jdedwards.base.logging.log4j.DefaultFormat
```

- File handler setting for jasdebug log
jdelog.Debug=DEBUG, jasdebug
jdelog.handler.jasdebug=com.jdedwards.base.logging.log4j.FileHandler
jdelog.handler.jasdebug.File=\\jasdebug.log
jdelog.handler.jasdebug.Level=DEBUG

See Also

- *Log4j Project* (<http://jakarta.apache.org/log4j/docs/index.html>) on the Apache Jakarta Project Web site more information about the Log4j Project

iJDEScript

GenCOM and GenJava use a scripting language called iJDEScript that allows you to script code generation activities. Other than a few small differences, the scripting language is the same for these generators. You can use iJDEScript to:

- Rename business function libraries or select different business functions to create a custom interface; for example:

```
library MyTestLibrary
```

```
interface MytestInterface
```

```
import B4200310 F4211FSEditLine
```

```
import B000042
```

This example selects the single business functions B4200310 F4211FSEditLine and B000042 for exposure.

- Use PeopleSoft EnterpriseOne object aliases for more meaningful names.
- Select PeopleSoft EnterpriseOne business functions to expose; for example:

```
library MyAnotherLibrary
```

```
importlib CAEC
```

```
importlib CRUNTIME 1
```

This example selects all of the business functions in the CAEC and CRUNTIME 1 libraries for exposure.

iJDEScript scripts have a simple syntax:

```
# comments begin with # and proceed to the end of line
```

```
# whitespace is ignored
```

```
login
```

```
importlib CAEC
```

```
build
```

iJDEScript Commands

iJDEScript supports a standard set of commands. These commands may vary slightly for GenCOM and GenJava. These variations are indicated in the command descriptions that follow.

Build Command

The build command tells the generator to generate code for all defined interfaces and build the appropriate libraries.

When the build command is complete, the interface definitions are released. Using the build command again only generates code for interfaces defined after the last build command.

Syntax

```
build
```

Call Command

The call command tells the generator to evaluate a subroutine with the given parameters. Parameters appear within the subroutine in order as special macros named %1%, %2%, and so on.

Syntax

```
call sub [param [...]]
```

Example

```
login
call GenerateLib CAEC
call GenerateLib CALLBSFN
build
logout
```

Define Command

The define command tells the generator to optionally define a macro expansion. The value is expanded first, and then stored as the expansion of macro name. If name already has an expansion, the generator ignores this command.

Syntax

```
define name value
```

Example

```
define val1 This is a test
define val2 %val1%!
define val2 This is ignored
say %val2%
generates the output
This is a test
```

Define! Command

The define! command tells the generator to define a macro expansion. The value is expanded first, and then stored as the expansion of macro name. If name already has an expansion, the generator replaces the current expansion with the new expansion.

Syntax

```
define name value
```

Example

```
define val1 This is a test
define val2 %val1%!
define! val2 This is not ignored
say %val2%
generates the output
This is not ignored
```

Exit Command

The exit command tells the generator to exit the current subroutine or command file.

Syntax

```
exit
```

Help Command

The help command requests help information from the generator on all available commands. Syntax information and a brief description are presented for each command. If command is specified, only help for command is shown.

Syntax

```
help [command]
```

Import Command

The import command tells the generator to retrieve the specification of a function or group of business functions from the database and add them to the current interface definition. If only the business-function name is specified, all functions from the specified business-function are retrieved and added to the current interface definition. If a function name is specified, only that function is retrieved and added to the current interface definition.

The alias option allows you to rename the function within the interface definition. The implementation still uses the original name when invoking the business function; however, the function is exposed as name through the interface.

Syntax

```
import business-function [function [alias name]]
```

Example

```
library General
interface ReleaseMgmt
# Load GetReleaseAndVersion from B9800890; call it GetRV in
# ReleaseMgmt
import B4200310 F4211FSEditLine alias GetRV
# Load all functions from B000042
import B000042
```

Importlib Command

The importlib command tells the generator to import all business functions from the specified PeopleSoft EnterpriseOne library, such as CAEC or CALLBSFN, into the current library definition. Each business function group results in the definition of an interface with the same name as the business function group and exposes as methods the functions within that group.

The category parameters allow the user to restrict the import to one or more specific categories (1, 2, 3 and -; see the /Cat command line option).

Syntax

```
importlib library [category [...]]
```

Example

```
library JDECOMInterfaceCAECCat1
# Load all category 1 functions from CAEC
importlib CAEC 1
build
```

Interface Command

The interface command tells the generator to begin the definition of an interface. All business functions retrieved using subsequent import commands become members of this interface.

Syntax for COM

```
interface interface [ProgID prog-id] [vi-prog-id]
```

COM Example

```
interface ReleaseMgmt ProgID SOA.ReleaseMgmt.5 SOA.ReleaseMgmt
import B4200310 F4211FSEditLine
```

Library Command

The library command tells the generator that subsequent interface and import commands will generate definitions that belong in the library (DLL) named *name*. If the parameterset tag is also supplied, the library is used solely for parameterset definitions.

Note

When the library command without the parameter set tag is evaluated, parametersets for subsequent interface and import commands appear in that library until a library command with the parameterset tag is evaluated.

Syntax

```
library name [parameterset]
```

Example:

```
library Lib1
library Lib1Params parameterset
# Parametersets for CALLBSFN go in Lib1Params, but the
# business function interfaces go in Lib1
importlib CALLBSFN 2 3
```

Login Command

The login command tells the generator to log on to PeopleSoft EnterpriseOne. If user, password, environment, and role are not specified, the user is prompted for the information.

Syntax

```
login [user password environment role]
```

Example

```
login me mypassword demo
```

Logout Command

The logout command tells the generator to log off of PeopleSoft EnterpriseOne.

Syntax

```
logout
```

Opt Command

The opt command tells the generator to set the value of a generator command line parameter. The option parameter should not begin with the usual "/". The value parameter does not undergo macro expansion.

Syntax

```
opt option value
```

Example

```
# Do not generate business function interfaces, only
# parameterset interfaces
opt NoBSFN
```

Rename Command

The rename command tells the generator to rename an interface or a method within an interface. If a method is renamed, the correct business function is still called to build the implementation, but the method is exposed through the interface with a different name.

Syntax

```
rename interface new
rename interface method new
```

Example

```
library Lib1
importlib CALLBSFN
rename B000042 BatchControl
rename BatchControl FSOpenBatch Open
rename BatchControl FSCloseBatch Close
```

Say Command

The say command tells the generator to display *message* on the console.

Syntax

```
say message
```

Example

```
say This is a test (%OwRelease%)
generate the output
This is a test (B9)
```

Sub Command

The sub command creates a subroutine definition. The call command may be used to invoke the subroutine. Parameters passed to the subroutine are as special macros named %1%, %2%, and so on.

Syntax

```
sub name
  commands
end
```

Example

```
sub GenerateLibrary
  define source %1%
  library JDECOMInterface%source%Cat1
  importlib %source% 1
  # Create a library of all category 2 business functions in source
  opt NoBSFN
  library JDECOMInterface%source%Cat2
  importlib %source% 2
  # Create a library of all category 3 business functions in
  source
```

```

        library JDECOMInterface%source%Cat3
importlib %source% 3
        system del /q c:\temp\*.*
        build
        # Move the libraries to a staging area
        system mkdir d:\build
        system mkdir d:\build\Cat1
system mkdir d:\build\Cat2
system mkdir d:\build\Cat3
system move JDECOMInterface%source%Cat1.* d:\build\Cat1
system move JDECOMInterface%source%Cat2.* d:\build\Cat2
system move JDECOMInterface%source%Cat3.* d:\build\Cat3
end
call GenerateLibrary CAEC

```

System Command

The system command tells the generator to evaluate command in the shell.

Syntax

```
system command
```

Example

```

say This is a test
generates the output
This is a test

```

EnterpriseOne PeopleBooks Glossary

“as of” processing	A process that is run at a specific point in time to summarize item transactions.
52 period accounting	A method of accounting that uses each week as a separate accounting period.
account site	In the invoice process, the address to which invoices are mailed. Invoices can go to a different location or account site from the statement.
active window	The window that contains the document or display that will be affected by current cursor movements, commands, and data entry in environments that are capable of displaying multiple on-screen windows.
ActiveX	A technology and set of programming tools developed by Microsoft Corporation that enable software components written in different languages to interact with each another in a network environment or on a web page. The technology, based on object linking and embedding, enables Java applet-style functionality for Web browsers as well as other applications (Java is limited to Web browsers at this time). The ActiveX equivalent of a Java applet is an ActiveX control. These controls bring computational, communications, and data manipulation power to programs that can “contain” them—for example, certain Web browsers, Microsoft Office programs, and anything developed with Visual Basic or Visual C++.
activity	In Advanced Cost Accounting, an aggregation of actions performed within an organization that is used in activity-based costing.
activity driver	A measure of the frequency and intensity of the demands that are placed on activities by cost objects. An activity driver is used to assign costs to cost objects. It represents a line item on the bill of activities for a product or customer. An example is the number of part numbers, which is used to measure the consumption of material-related activities by each product, material type, or component. The number of customer orders measures the consumption of order-entry activities by each customer. Sometimes an activity driver is used as an indicator of the output of an activity, such as the number of purchase orders that are prepared by the purchasing activity. See also cost object.
activity rule	The criteria by which an object progresses from a given point to the next in a flow.
actual cost	Actual costing uses predetermined cost components, but the costs are accumulated at the time that they occur throughout the production process.
adapter	A component that connects two devices or systems, physically or electronically, and enables them to work together.
add mode	The condition of a form where a user can enter data into it.
advanced interactive executive	An open IBM operating system that is based on UNIX.
agent	A program that searches through archives or other repositories of information on a topic that is specified by the user.

aging	A classification of accounts by the time elapsed since the billing date or due date. Aging is divided into schedules or accounting periods, such as 0-30 days, 31-60 days, and so on.
aging schedule	A schedule that is used to determine whether a payment is delinquent and the number of days which the payment is delinquent.
allegato IVA clienti	In Italy, the term for the A/R Annual VAT report.
allegato IVA fornitori	In Italy, the term for the A/P Annual VAT report.
application layer	The seventh layer of the Open Systems Interconnection Reference Model, which defines standards for interaction at the user or application program level.
application programming interface (API)	A set of routines that is used by an application program to direct the performance of procedures by the computer's operating system.
AS/400 Common	A data source that resides on an AS/400 and holds data that is common to the co-existent library, allowing PeopleSoft EnterpriseOne to share information with PeopleSoft World.
assembly inclusion rule	A logic statement that specifies the conditions for using a part, adjusting the price or cost, performing a calculation, or using a routing operation for configured items.
audit trail	The detailed, verifiable history of a processed transaction. The history consists of the original documents, transaction entries, and posting of records and usually concludes with a report.
automatic return	A feature that allows a user to move to the next entry line in a detail area or to the first cell in the next row in several applications.
availability	The expression of the inventory amount that can be used for sales orders or manufacturing orders.
available inventory	The quantity of product that can be promised for sale or transfer at a particular time, considering current on-hand quantities, replenishments in process, and anticipated demand.
back office	The set of enterprise software applications that supports the internal business functions of a company.
backhaul	The return trip of a vehicle after delivering a load to a specified destination. The vehicle can be empty or the backhaul can produce less revenue than the original trip. For example, the state of Florida is considered a backhaul for many other states—that is, many trucking companies ship products into the state of Florida, but most of them cannot fill a load coming out of Florida or they charge less. Hence, trucks coming out of Florida are either empty or produce less revenue than the original trip.
balance forward	The cumulative total of inventory transactions that is used in the Running Balance program. The system does not store this total. You must run this program each time that you want to review the cumulative inventory transactions total.

balance forward receipt application method	A receipt application method in which the receipt is applied to the oldest or newest invoices in chronological order according to the net due date.
bank tape (lock box) processing	The receipt of payments directly from a customer's bank via customer tapes for automatic receipt application.
base location	[In package management] The topmost location that is displayed when a user launches the Machine Identification application.
basket discount	A reduction in price that applies to a group or "basket" of products within a sales order.
basket repricing	A rule that specifies how to calculate and display discounts for a group of products on a sales order. The system can calculate and display the discount as a separate sales order detail line, or it can discount the price of each item on a line-by-line basis within the sales order.
batch job	A job submitted to a system and processed as a single unit with no user interaction.
batch override	An instruction that causes a batch process to produce output other than what it normally would produce for the current execution only.
batch process	A type of process that runs to completion without user intervention after it has been started.
batch program	A program that executes without interacting with the user.
batch version	A version of a report or application that includes a set of user-defined specifications, which control how a batch process runs.
batch/lot tracking	The act of identifying where a component from a specific lot is used in the production of goods.
batch/mix	A manufacturing process that primarily schedules short production runs of products.
batch-of-one processing	A transaction method that allows a client application to perform work on a client workstation, and then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks. See also direct connect, store-and-forward.
binary large object (BLOB)	A collection of binary data stored as a single entity in a [file].
binder clip	See paper clip.
black products	Products that are derived from the low or heavy end of the distillation process—for example, diesel oils and fuel oils. See also white products.
blend note	Document that authorizes a blending activity, and describes both the ingredients for the blend and the blending steps that occur.

blend off	Reworking off-specification material by introducing a small percentage back into another run of the same product.
blind execution	The mode of execution of a program that does not require the user to review or change the processing options set for the program, and does not require user intervention after the program has been launched.
boleto	In Brazil, the document requesting payment by a supplier or a bank on behalf of a supplier.
bolla doganale	VAT-Only Vouchers for Customs. In Italy, a document issued by the customs authority to charge VAT and duties on extra-EU purchasing.
bookmark	A shortcut to a location in a document or a specific place in an application or application suite.
bordero & cheque	In Brazil, bank payment reports.
broker	A program that acts as an intermediary between clients and servers to coordinate and manage requests.
BTL91	In the Netherlands, the ABN/AMRO electronic banking file format that enables batches with foreign automatic payment instructions to be delivered.
budgeted volume	A statement of planned volumes (capacity utilization) upon which budgets for the period have been set.
bunkering	A rate per ton or a sum of money that is charged for placing fuel on board; can also mean the operation itself.
business function	An encapsulated set of business rules and logic that can normally be re-used by multiple applications. Business functions can execute a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the APIs that allow them to be called from a form, a database trigger, or a non-EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.
business function event rule	Encapsulated, reusable business logic that is created by using through event rules rather than C programming. Contrast with embedded event rule. See also event rule.
business object library	[In interoperability] The repository that stores EnterpriseOne business objects, which consist of Java or CORBA objects.
business unit	A financial entity that is used to track the costs, revenue, or both, of an organization. A business unit can also be defined as a branch/plant in which distribution and manufacturing activities occur. Additionally, in manufacturing setup, work centers and production lines must be defined as business units; but these business unit types do not have profit/loss capability.

business view	Used by EnterpriseOne applications to access data from database tables. A business view is a means for selecting specific columns from one or more tables with data that will be used in an application or report. It does not select specific rows and does not contain any physical data. It is strictly a view through which data can be handled.
business view design aid (BDA)	An EnterpriseOne GUI tool for creating, modifying, copying, and printing business views. The tool uses a graphical user interface.
buy-back crude	In foreign producing oil countries, that portion of the host government's share of "participation crude" which it permits the company holding a concession to "buy back."
CAB	In Italy, the bank branch code or branch ID. A five-digit number that identifies any agency of a specific bank company in Italy.
cadastro de pessoas fisicas	Cadastro de pessoas físicas. In Brazil, the federal tax ID for a person.
category code	A code that identifies a collection of objects sharing at least one common attribute.
central object	A software component that resides on a central server.
central objects merge	A process that blends a customer's modifications with the objects in a current release with objects in a new release.
central server	A computer that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers.
certificate input	See direct input.
certificate of analysis (COA)	A document that is a record of all of the testing which has been performed against an item, lot, or both, plus the test results for that item and lot.
change management	[In software development] A process that aids in controlling and tracking the evolution of software components.
change order	In PeopleSoft, an addendum to the original purchase order that reflects changes in quantities, dates, or specifications in subcontract-based purchasing. A change order is typically accompanied by a formal notification.
chargeback	A receipt application method that generates an invoice for a disputed amount or for the difference of an unpaid receipt.
chart	EnterpriseOne term for tables of information that appear on forms in the software. See forms.
check-in location	The directory structure location for the package and its set of replicated objects. This location is usually \\deploymentserver\release\path_code\package\packagename. The subdirectories under this path are where the central C components (source, include, object, library, and DLL file) for business functions are stored.

checksum value	A computed value that depends on the contents of a block of data, and that is transmitted or stored with the data to detect whether errors have occurred in the transmission or storage.
class	[In object-oriented programming] A category of objects that share the same characteristics.
clean cargo	Term that refers to cargoes of gasoline and other refined products. See also dirty cargo.
client access	The ability to access data on a server from a client machine.
client machine	Any machine that is connected to a network and that exchanges data with a server.
client workstation	A network computer that runs user application software and is able to request data from a server.
ClieOp03	In the Netherlands, the euro-compliant uniform electronic banking file format that enables batches with domestic automatic direct debit instructions and batches with domestic payment instructions to be delivered.
ClieOp2	In the Netherlands, the uniform electronic banking file format that enables batches with domestic automatic direct debit instructions and batches with domestic payment instructions to be delivered.
cluster	Two or more computers that are grouped together in such a way that they behave like a single computer.
co-existence	A condition where two or more applications or application suites access one or more of the same database tables within the same enterprise.
cold test	The temperature at which oil becomes solid. Generally considered to be 5 degrees F lower than the pour point.
commitment	The number of items that are reserved to fill demand.
common object request broker architecture	An object request broker standard that is endorsed by the Object Management Group.
compa-ratio	An employee's salary divided by the midpoint amount for the employee's pay grade.
component changeout	See component swap.
component object model (COM)	A specification developed by Microsoft for building software components that can be assembled into programs or add functionality to existing programs running on Microsoft Windows platforms. COM components can be written in a variety of languages, although most are written in C++, and can be unplugged from a program at runtime without having to recompile the program.

component swap	In Equipment/Plant Management, the substitution of an operable component for one that requires maintenance. Typically, you swap components to minimize equipment downtime while servicing one of the components. A component swap can also mean the substitution of one parent or component item for another in its associated bill of material.
conference room pilot environment	An EnterpriseOne environment that is used as a staging environment for production data, which includes constants and masters tables such as company constants, fiscal date patterns, and item master. Use this environment along with the test environment to verify that your configuration works before you release changes to end-users.
configurable network computing (CNC)	An application architecture that allows interactive and batch applications that are composed of a single code base to run across a TCP/IP network of multiple server platforms and SQL databases. The applications consist of re-usable business functions and associated data that can be configured across the network dynamically. The overall objective for businesses is to provide a future-proof environment that enables them to change organizational structures, business processes, and technologies independently of each other.
configurable processing engine	Handles all “batch” processes, including reporting, Electronic Data Exchange (EDI) transactions, and data duplication and transformation (for data warehousing). This ability does not mean that it exists only on the server; it can be configured to run on desktop machines (Windows 95 and NT Workstation) as well.
configuration management	A rules-based method of ordering assemble-to-order or make-to-order products in which characteristics of the product are defined as part of the Sales Order Entry process. Characteristics are edited by using Boolean logic, and then translated into the components and routing steps that are required to produce the product. The resulting configuration is also priced and costed, based on the defined characteristics.
configured item segment	A characteristic of a configured item that is defined during sales order entry. For example, a customer might specify a type of computer hard drive by stating the number of megabytes of the hard drive, rather than a part number.
consuming location	The point in the manufacturing routing where a component or subassembly is used in the production process. In kanban processing, the location where the kanban container materials are used in the manufacturing process and the kanban is checked out for replenishment.
contra/clearing account	A G/L account used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations.
contribution to profit	Selling price of an item minus its variable costs.
control table	A table that controls the program flow or plays a major part in program control.
control table workbench	During the Installation Workbench process, Control Table Workbench runs the batch applications for the planned merges that update the data dictionary, user defined codes, menus, and user overrides tables.

control tables merge	A process that blends a customer's modifications to the control tables with the data that accompanies a new release.
corrective work order	A work order that is used to formally request unscheduled maintenance and communicate all of the details pertaining to the requested maintenance task.
corrective work order	A work order that is used to formally request unscheduled maintenance and communicate all of the details pertaining to the requested maintenance task.
cost assignment	Allocating resources to activities or cost objects.
cost component	An element of an item's cost—for example, material, labor, or overhead.
cost object	Any customer, product, service, contract, project, or other work unit for which you need a separate cost measurement.
cost rollup	A simulated scenario in which work center rates, material costs, and labor costs are used to determine the total cost of an item.
costing elements	The individual classes of added value or conversion costs. These elements are typically materials, such as raw and packaging; labor and machine costs; and overhead, such as fixed and variable. Each corporation defines the necessary detail of product costs by defining and tracking cost categories and subcategories.
credit memo	A negative amount that is used to correct a customer's statement when he or she is overcharged.
credit notice	The physical document that is used to communicate the circumstances and value of a credit order.
credit order	A credit order is used to reflect products or equipment that is received or returned so that it can be viewed as a sales order with negative amounts. Credit orders usually add the product back into inventory. This process is linked with delivery confirmation.
cross segment edit	A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced.
crude oil assay	A procedure for determining the distillation curve and quality characteristics of a crude oil.
cumulative update	A version of software that includes fixes and enhancements that have been made since the last release or update.
currency relationships	When converting amounts from one currency to another, the currency relationship defines the from currency and the to currency in PeopleSoft software. For example, to convert amounts from German marks to the euro, you first define a currency relationship between those two currencies.
currency restatement	The process of converting amounts from one currency into another currency, generally for reporting purposes. It can be used, for example, when many currencies must be restated into a single currency for consolidated reporting.

current cost	The cost that is associated with an item at the time a parts list and routing are attached to a work order or rate schedule. Current cost is based on the latest bill of material and routing for the item.
customer pricing rules	In Procurement, the inventory pricing rules that are assigned to a supplier. In Sales, inventory pricing rules that are assigned to a customer.
D.A.S. 2 Reporting (DAS 2 or DADS 1)	In France, the name of the official form on which a business must declare fees and other forms of remuneration that were paid during the fiscal year.
data dictionary	A dynamic repository that is used for storing and managing a specific set of data item definitions and specifications.
data source workbench	During the Installation Workbench process, Data Source Workbench copies all of the data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the System - release number data source. It also updates the Data Source Plan detail record to reflect completion.
data structure	A description of the format of records in a database such as the number of fields, valid data types, and so on.
data types	Supplemental information that is attached to a company or business unit. Narrative type contains free-form text. Code type contains dates, amounts, and so on.
datagram	A self-contained packet of information that is forwarded by routers, based on their address and the routing table information.
date pattern	A period of time that is set for each period in standard and 52-period accounting and forecasting.
DCE	See distributed computing environment.
DEB	See déclaration d'échange de biens.
debit memo	In Accounts Payable, a voucher that is entered with a negative amount. Enter this type of voucher when a supplier sends you a credit so that you can apply the amount to open vouchers when you issue payment to the supplier.
debit memo	A form that is issued by a customer, requesting an adjustment of the amount, which is owed to the supplier.
debit statement	A list of debit balances.
de-blend	When blend off does not result in a product that is acceptable to customers. The further processing of product to adjust specific physical and chemical properties to within specification ranges. See also blend off.
déclaration d'échange de biens (DEB)	The French term that is used for the Intrastat report.
delayed billing	The invoicing process is delayed until the end of a designated period.

delta load	A batch process that is used to compare and update records between specified environments.
denominated-in currency	The company currency in which financial reports are based.
deployment server	A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.
detail	The specific information that makes up a record or transaction. Contrast with summary.
detail information	Information that primarily relates to individual lines in a sales or purchase order.
direct connect	A transaction method in which a client application communicates interactively and directly with a server application. See also batch-of-one immediate, store-and-forward.
direct input	The system calculates the net units when you enter gross volume, temperature, and gravity or density. This data is generally entered during product receiving from the certificate that is prepared by an independent inspector.
direct ship orders	A purchase order that is issued to a third-party supplier who designates the destination as the customer. A direct ship sales order is also created for the customer. Direct ship orders occur when a product is not available from a company-owned or company-operated source, so the system creates an order to ship the product from a third-party source directly to the customer. Sometimes referred to as a drop ship or third-party supply.
direct usage	Consumption of resources that are attributable to specific production runs because the resources were directly issued to the schedule/order.
director	An EnterpriseOne user interface that guides a user interactively through an EnterpriseOne process.
dirty cargo	Term that refers to crude oil cargoes or other non-refined petroleum cargoes. See also clean cargo.
dispatch planning	Efficient planning and scheduling of product deliveries. Considerations include: Dispatch groups Scheduled delivery date Scheduled delivery time Preferred delivery date Preferred delivery time Average delivery time for that geographical location Available resources Special equipment requirements at the product's source or destination.

displacement days	The number of days that are calculated from today's date by which you group vouchers for payment. For example, if today's date is March 10 and you specify three displacement days, the system includes vouchers with a due date through March 13 in the payment group. Contrast with pay-through date.
display sequence	A number that the system uses to re-order a group of records on the form.
distributed computing environment (DCE)	A set of integrated software services that allows software which is running on multiple computers to perform seamless and transparently to the end-users. DCE provides security, directory, time, remote procedure calls, and files across computers running on a network.
distributed data processing	Processing in which some of the functions are performed across two or more linked facilities or systems.
distributed database management system (DDBMS)	A system for distributing a database and its control system across many geographically dispersed machines.
do not translate (DNT)	A type of data source that must exist on the AS/400 because of BLOB restrictions.
double-byte character set (DBCS)	A method of representing some characters by using one byte and other characters by using two bytes. Double-byte character sets are necessary to represent some characters in the Japanese, Korean, and Chinese languages.
downgrade profile	A statement of the hierarchy of allowable downgrades. Includes substitutions of items, and meeting tighter specifications for those products with wider or overlapping specification ranges.
DTA	Datenträgeraustausch. A Swiss payment format that is required by Telekurs (Payserv).
dual pricing	To provide prices for goods and services in two currencies. During the euro transition period, dual pricing between the euro and Economic and Monetary Union (EMU) member currencies is encouraged.
dynamic link library (DLL)	A set of program modules that are designed to be invoked from executable files when the executable files are run, without having to be linked to the executable files. They typically contain commonly used functions.
dynamic partitioning	The ability to dynamically distribute logic or data to multiple tiers in a client/server architecture.
economy of scale	A phenomenon whereby larger volumes of production reduce unit cost by distributing fixed costs over a larger quantity. Variable costs are constant; but fixed costs per unit are reduced, thereby reducing total unit cost.
edit mode	A processing mode or condition where the user can alter the information in a form.
edit rule	A method that is used for formatting user entries, validating user entries, or both, against a predefined rule or set of rules.

embedded event rule	An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field that is based on a processing option value, or calling a business function. Contrast with business function event rule. See also event rule.
employee work center	A central location for sending and receiving all EnterpriseOne messages (system and user-generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages. With respect to workflow, the Message Center is MAPI compliant and supports drag-and-drop work reassignment, escalation, forward and reply, and workflow monitoring. All messages from the message center can be viewed through EnterpriseOne messages or Microsoft Exchange.
Emulator	An item of software or firmware that allows one device to imitate the functioning of another.
encapsulation	The ability to confine access to and manipulation of data within an object to the procedures that contribute to the definition of that object.
engineering change order (ECO)	A work order document that is used to implement and track changes to items and resulting assemblies. The document can include changes in design, quantity of items required, and the assembly or production process.
enhanced analysis database	A database containing a subset of operational data. The data on the enhanced analysis database performs calculations and provides summary data to speed generation of reports and query response times. This solution is appropriate when external data must be added to source data, or when historical data is necessary for trend analysis or regulatory reporting. See also duplicated database, enterprise data warehouse.
enterprise server	A computer containing programs that collectively serve the needs of an enterprise rather than a single user, department, or specialized application.
EnterpriseOne object	A re-usable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects. See also object.
EnterpriseOne process	Allows EnterpriseOne clients and servers to handle processing requests and execute transactions. A client runs one process, and servers can have multiple instances of a process. EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes do not have to wait if the server is particularly busy.
EnterpriseOne web development computer	A standard EnterpriseOne Windows developer computer with the additional components installed: Sun's JDK 1.1. JFC (0.5.1). Generator Package with Generator.Java and JDECOM.dll. R2 with interpretive and application controls/form.

environment workbench	During the Installation Workbench process, Environment Workbench copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the System release number data source. It also updates the Environment Plan detail record to reflect completion.
equivalent fuel	A barrel of equivalent fuel supplies six million BTUs of heat. Fuel gas quantities are usually calculated as equivalent fuel barrels in economic calculations for refinery operations.
escalation monitor	A batch process that monitors pending requests or activities, and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time.
ESR	Einzahlungsschein mit Referenznummer. A pay slip with a reference number.
event rule	[In EnterpriseOne] A logic statement that instructs the system to perform one or more operations that are based on an activity that can occur in a specific application, such as entering a form or exiting a field.
exit bar	[In EnterpriseOne] The tall pane with icons in the left portion of many EnterpriseOne program windows.
facility	An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. Sometimes referred to as a business unit.
fast path	[In EnterpriseOne] A command prompt that allows the user to move quickly among menus and applications by using specific commands.
file handle	A temporary reference (typically a number) that is assigned to a file which has been opened by the operating system and is used throughout the session to access the file.
file server	A computer that stores files to be accessed by other computers on the network.
find/browse	A type of form used to: Search, view, and select multiple records in a detail area. Delete records. Exit to another form. Serve as an entry point for most applications.
firm planned order (FPO)	A work order that has reached a user defined status. When this status is entered in the processing options for the various manufacturing programs, messages for those orders are not exploded to the components.
fiscal date pattern	A representation of the beginning date for the fiscal year and the ending date for each period in that year.
fix/inspect	A type of form used to view, add, or modify existing records. A fix/inspect form has no detail area.

fixed quantity	A term that indicates the bill of material relationship between a parent item and its components or ingredients. When a bill of material component has a fixed quantity relationship to its parent, the amount of the component does not change when the software calculates parts list requirements for different work order quantities. Contrast with variable quantity.
flexible account numbers	The format of account numbers for journal entries. The format that you set up must be the three segments: Business unit. Object. Subsidiary.
form design aid (FDA)	The EnterpriseOne GUI development tool for building interactive applications and forms.
form exit	[In EnterpriseOne] An option that is available as a button on the Form Exit bar or as a selection in the Form menu. It allows users to open an interconnected form.
form interconnection	Allows one form to access and pass data to another form. Form interconnections can be attached to any event; however, they are normally used when a button is clicked.
form type	The following form types are available in EnterpriseOne: Find/browse. Fix/inspect. Header detail. Headerless detail. Message. Parent/child. Search/select.
form-to-form call	A request by a form for data or functionality from one of the connected forms.
framework	[In object-oriented systems] A set of object classes that provide a collection of related functions for a user or piece of software.
frozen cost	The cost of an item, operation, or process after the frozen update program is run; used by the Manufacturing Accounting system.
frozen update program	A program that freezes the current simulated costs, thereby finalizing them for use by the Manufacturing Accounting system.
globally unique identifier (GUI)	A 16-byte code in the Component Object Model that identifies an interface to an object across all computers and networks.
handle	[In programming] A pointer that contains the address of another pointer, which, in turn, contains the address of the desired object.

hard commitment	The number of items that are reserved for a sales order, work order, or both, from a specific location, lot, or both.
hard error	An error that cannot be corrected by a given error detection and correction system.
header	Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows.
header information	Information that pertains to the entire order.
hover help	A help function that provides contextual information or instructions when a cursor moves over a particular part of the interface element for a predefined amount of time.
ICMS	Imposto sobre circulação de mercadoria e serviços. In Brazil, a state tax that is applied to the movement of merchandise and some services.
ICMS Substituto	Imposto sobre circulação de mercadoria e serviços substituto. In Brazil, the ICMS tax that is charged on interstate transactions, or on special products and clients.
ICMS Substituto-Markup	See imposto sobre circulação de mercadoria e serviços substituto-markup.
imposto de renda (IR)	Brazilian income tax.
imposto sobre produtos industrializados	In Brazil, a federal tax that applies to manufactured goods (domestic and imported).
imposto sobre services (ISS)	In Brazil, tax on services.
inbound document	A document that is received from a trading partner using Electronic Data Interface (EDI). This document is also referred to as an inbound transaction.
indented tracing	Tracking all lot numbers of intermediates and ingredients that are consumed in the manufacture of a given lot of product, down through all levels of the bill of material, recipe, or formula.
indexed allocations	A procedure that allocates or distributes expenses, budgets, adjustments, and so on, among business units, based on a fixed percentage.
indirect measurement	Determining the quantity on-hand by: Measuring the storage vessels and calculating the content's balance quantity. or Theoretically calculating consumption of ingredients and deducting them from the on-hand balance.
indirect usage	Determining what should have been used by multiplying receipt quantity of the parent times the quantity per statement in the formula, recipe, or bill of material. This transaction typically affects both consumption on schedule as well as issue from on-hand balances.

in-process rework	Recycling a semi processed product that does not meet acceptable standards. Further processing takes the product out of a given operation and sends it back to the beginning of that operation or a previous operation (for example, unreacted materials). Rework that is detected prior to receipt of finished goods and corrected during the same schedule run.
INPS withholding tax	Instituto Nazionale di Previdenza Sociale withholding tax. In Italy, a 12% social security withholding tax that is imposed on payments to certain types of contractors. This tax is paid directly to the Italian social security office.
inscrição estadual	ICMS tax ID. In Brazil, the state tax ID.
inscrição municipal	ISS tax ID. In Brazil, the municipal tax ID.
integrated toolset	Unique to EnterpriseOne is an industrial-strength toolset that is embedded in the already comprehensive business applications. This toolset is the same toolset that is used by PeopleSoft to build EnterpriseOne interactive and batch applications. Much more than a development environment, however, the EnterpriseOne integrated toolset handles reporting and other batch processes, change management, and basic data warehousing facilities.
integrity test	A process that is used to supplement a company's internal balancing procedures by locating and reporting balancing problems and data inconsistencies.
interbranch sales order	A sales order that is used for transactions between branch/plants other than the selling branch/plant.
Interoperability	The ability of different computer systems, networks, operating systems, and applications to work together and share information.
inventory pricing rule	A discount method that is used for purchases from suppliers and sales to customers. The method is based on effectivity dates, up-to quantities, and a factor by which you can mark up or discount the price or cost.
inventory turn	The number of times that the inventory cycles, or turns over, during the year. A frequently used method to compute inventory turnover is to divide the annual costs of sales by the average inventory level.
invoice	An itemized list of goods that are shipped or services that are rendered, stating quantities, prices, fees, shipping charges, and so on. Companies often have their invoices mailed to a different address than where they ship products. In such cases, the bill-to address differs from the ship-to address.
IP	See imposto sobre produtos industrializados.
IR	See imposto de renda.
IServer Service	Developed by PeopleSoft, this Internet server service resides on the Web server and is used to speed up delivery of the Java class files from the database to the client.
ISS	See imposto sobre servicos.

jargon	An alternate data dictionary item description that EnterpriseOne or PeopleSoft World displays, based on the product code of the current object.
java application server	A component-based server that resides in the middle-tier of a server-centric architecture and provides middleware services for security and state maintenance, along with data access and persistence.
JDBNET	A database driver that allows heterogeneous servers to access each other's data.
jde.ini	A PeopleSoft file (or member for AS/400) that provides the runtime settings that are required for EnterpriseOne initialization. Specific versions of the file or member must reside on every machine that is running EnterpriseOne, including workstations and servers.
JDE.LOG	The main diagnostic log file of EnterpriseOne. Always located in the root directory on the primary drive. Contains status and error messages from the startup and operation of EnterpriseOne.
JDEBASE Database Middleware	PeopleSoft proprietary database middleware package that provides two primary benefits: 1. Platform-independent APIs for multidatabase access. These APIs are used in two ways: a. By the interactive and batch engines to dynamically generate platform-specific SQL, depending on the data source request. b. As open APIs for advanced C business function writing. These APIs are then used by the engines to dynamically generate platform-specific SQL. 2. Client-to-server and server-to-server database access. To accomplish this access, EnterpriseOne is integrated with a variety of third-party database drivers, such as Client Access 400 and open database connectivity (ODBC).
JDECallObject	An application programming interface that is used by business functions to invoke other business functions.
JDEIPC	Communications programming tools that are used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes.
JDENET	PeopleSoft proprietary middleware software. JDENET is a messaging software package.
JDENET communications middleware	PeopleSoft proprietary communications middleware package for EnterpriseOne. It is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all EnterpriseOne supported platforms.
just in time installation (JITI)	EnterpriseOne's method of dynamically replicating objects from the central object location to a workstation.
just in time replication (JITR)	EnterpriseOne's method of replicating data to individual workstations. EnterpriseOne replicates new records (inserts) only at the time that the user needs the data. Changes, deletes, and updates must be replicated using Pull Replication.

Kagami	In Japan, summarized invoices that are created monthly (in most cases) to reduce the number of payment transactions.
latitude	The X coordinate of the location of an item in the warehouse. The system can use latitude, longitude, and height when suggesting locations for putaway, replenishment, and picking.
laytime (or layhours)	The amount of time that is allotted to a tanker at berth to complete loading or discharging cargo. This time is usually expressed in running hours, and is fixed by prior agreement between the vessel owner and the company that is chartering the vessel. Laytime is stipulated in the charter, which states exactly the total of number of hours that are granted at both loading and unloading ports, and indicates whether such time is reversible. A statement of “Seventy-Two Hours, Reversible” means that a total of 72 hours is granted overall at both ports, and any time saved at one port can be applied as a credit at the other port. For example, if the vessel uses only 32 hours instead of 36 hours to load cargo, it can apply an additional four hours to the 36 hours allotted at the discharge port. Such considerations are important for purposes of computing demurrage.
leading zeros	A series of zeros that certain facilities in PeopleSoft systems place in front of a value that is entered. This situation normally occurs when you enter a value that is smaller than the specified length of the field. For example, if you enter 4567 in a field that accommodates eight numbers, the facility places four zeros in front of the four numbers that you enter. The result appears as 00004567.
ledger type	A code that designates a ledger which is used by the system for a particular purpose. For example, all transactions are recorded in the AA (actual amounts) ledger type in their domestic currency. The same transactions can also be stored in the CA (foreign currency) ledger type.
level break	The position in a report or text where a group of similar types of information ends and another one begins.
libro IVA	Monthly VAT report. In Italy, the term for the report that contains the detail of invoices and vouchers that were registered during each month.
line of business	A description of the nature of a company’s work; also a tool to control the relationship with that customer, including product pricing.
linked service type	A service type that is associated with a primary service type. Linked service types can be cancelled, and the maintenance tasks are performed when the primary service type to which they are linked comes due. You can specify whether the system generates work orders for linked service types, as well as the status that the system assigns to work orders that have already been generated. Sometimes referred to as associated service types. See also primary service type and service type.
livro razao	In Brazil, a general ledger report.
load balancing	The act of distributing the number of processes proportionally to all servers in a group to maximize overall performance.

location workbench	During the Installation Workbench process, Location Workbench copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the System data source.
log files	Files that track operations for a process or application. Reviewing log files is helpful for troubleshooting problems. The file extension for log files is .LOG.
logic data source	Any code that provides data during runtime.
logical compartment	One of two ways that is identified in the transportation constants to display compartments on vehicles. Logical display numbers the compartments sequentially. For example, if two vehicles are on a trip and each vehicle has three compartments, the logical display is 1,2,3,4,5,6.
logical file	A set of keys or indices that is used for direct access or ordered access to the records in a physical file. Several logical files can have different accesses to a physical.
logical shelf	A logical, not physical, location for inventory that is used to track inventory transactions in loan/borrow, or exchange agreements with other companies. See also logical warehouse.
logical warehouse	Not a physical warehouse containing actual inventory, but a means for storing and tracking information for inventory transactions in loan/borrow, or exchange agreements with other companies.
longitude	The Y coordinate of the location of an item in the warehouse. The system can use latitude, longitude, and height when suggesting locations for putaway, replenishment, and picking.
LSV	Lastschriftverfahren. A Swiss auto debit format that is required by Telekurs (Payserv).
mail merge	A mass-mail facility that takes names, addresses, and (sometimes) pertinent facts about recipients and merges the information into a form letter or a similarly basic document.
mailmerge workbench	[In EnterpriseOne] An application that merges Microsoft Word 6.0 (or higher) word-processing documents with EnterpriseOne records to automatically print business documents.
main fuels	Usually refers to bulk fuel products, but sometimes includes packaged products.
maintenance loop	See maintenance route.
maintenance route	A method of performing PMs for multiple pieces of equipment from a single preventive maintenance work order. A maintenance route includes pieces of equipment that share one or more identical maintenance tasks which can be performed at the same time for each piece of equipment. Sometimes referred to as maintenance loop.

maintenance work order	In PeopleSoft EnterpriseOne systems, a term that is used to distinguish work orders created for the performance of equipment and plant maintenance from other work orders, such as manufacturing work orders, utility work orders, and engineering change orders.
manufacturing and distribution planning	Planning that includes resource and capacity planning, and material planning operations. Resource and capacity planning allows you to prepare a feasible production schedule that reflects your demand forecasts and production capability. Material Planning Operations provides a short-range plan to cover material requirements that are needed to make a product.
mapping	A set of instructions that describes how one data structure passes data to another.
master business function	An interactive master file that serves as a central location for adding, changing, and updating information in a database.
master business function	A central system location for standard business rules about entering documents, such as vouchers, invoices, and journal entries. Master business functions ensure uniform processing according to guidelines that you establish.
master table	A database table that is used to store data and information that is permanent and necessary to the system's operation. Master tables might contain data such as paid tax amounts, supplier names, addresses, employee information, and job information.
matching document	A document that is associated with an original document to complete or change a transaction. For example, a receipt is the matching document of an invoice.
media object	An electronic or digital representation of an object.
media storage objects	Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.
memory violation	An error that occurs as the result of a memory leak.
menu selection	An option on a menu that initiates a software function directly.
message center	A central location for sending and receiving all EnterpriseOne messages (system- and user-generated), regardless of the originating application or user.
messaging application programming interface (MAPI)	An architecture that defines the components of a messaging system and how they behave. It also defines the interface between the messaging system and the components.
metal content	A series of properties of a blended product that help to determine its suitability for a prescribed purpose.
metals management	The process of maintaining information about the location and status of durable product containers such as liquid petroleum gas (LPG) cylinders.
mobile inventory	Inventory that is transferred from a depot to a barge or truck for milk-run deliveries.

modal	A restrictive or limiting interaction that is created by a given condition of operation. Modal often describes a secondary window that restricts a user's interaction with other windows. A secondary window can be modal with respect to its primary window or to the entire system. A modal dialog box must be closed by the user before the application continues.
model work order	For scheduled preventive maintenance or for a condition-based alert, a model work order functions as a template for the creation of other work orders. You can assign model work orders to service types and condition-based alerts. When the service type comes due or the alert is generated, the system automatically generates a work order that is based on information from the model work order.
modeless	Not restricting or limiting interaction. Modeless often describes a secondary window that does not restrict a user's interaction with other windows. A modeless dialog box stays on the screen and is available for use at any time, but also permits other user activities.
multiple stocking locations	Authorized storage locations for the same item number at locations, in addition to the primary stocking location.
multitier architecture	A client/server architecture that allows multiple levels of processing. A tier defines the number of computers that can be used to complete some defined task.
named event rules (NER)	Also called business function event rules. Encapsulated, re-usable business logic that is created by using event rules, rather than C programming.
national language support (NLS)	Mechanisms that are provided to facilitate internationalization of both system and application user interfaces.
natureza da operação	Transaction nature. In Brazil, a code that classifies the type of commercial transaction to conform to the fiscal legislation.
negative pay item	An entry in an account that indicates a prepayment. For example, you might prepay a supplier before goods are sent or prepay an employee's forecasted expenses for a business trip. The system stores these pending entries, assigning them a minus quantity as debit amounts in a designated expense account. After the prepaid goods are received or the employee submits an expense report, entering the actual voucher clears all of the negative pay items by processing them as regular pay items. Note that a negative pay item can also result from entering a debit memo (A/P) or a credit memo (A/R).
net added cost	The cost to manufacture an item at the current level in the bill of material. Thus, for manufactured parts, the net added cost includes labor, outside operations, and cost extras applicable to this level in the bill of material, but not materials (lower-level items). For purchased parts, the net added cost also includes the cost of materials.
next status	The next step in the payment process for payment control groups. The next status can be either WRT (write) or UPD (update).
node	A termination point for two or more communications links. A node can serve as the control location for forwarding data among the elements of a network or multiple networks, as well as performing other networking and, in some cases, local processing.

non-inventory items	See non-stock items.
non-list price	A price for bulk products that is determined by its own algorithms, such as a rolling average or commodity price plus.
non-prime product	A manufactured product with revenue potential that is less than the product planned for, or scheduled to be produced.
non-stock items	Items that the system does not account for as part of the inventory. For example, office supplies, or packaging materials can be non-stock items.
nota fiscal	In Brazil, a legal document that must accompany all commercial transactions.
nota fiscal fatura	In Brazil, a nota fiscal and invoice information.
notula	In Italy, the process whereby a business does not recognize value added tax until the payment of a voucher.
object configuration manager (OCM)	EnterpriseOne's object request broker and the control center for the runtime environment. It keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, the Object Configuration Manager directs access to it by using defaults and overrides for a given environment and user.
object embedding	When an object is embedded in another document, an association is maintained between the object and the application that created it; however, any changes made to the object are also only kept in the compound document. See also object linking.
object librarian	A repository of all versions, applications, and business functions that are re-usable in building applications.
object linking	When an object is linked to another document, a reference is created with the file in which the object is stored, as well as with the application that created it. When the object is modified, either from the compound document or directly through the file in which it is saved, the change is reflected in that application as well as anywhere it has been linked. See also object embedding.
object linking and embedding (OLE)	A technology for transferring and sharing information among applications by allowing the integration of objects from diverse applications, such as graphics, charts, spreadsheets, text, or an audio clip from a sound program. OLE is a compound document standard that was developed by Microsoft Corporation. It enables you to create objects with one application, and then link or embed them in a second application. Embedded objects retain their original format and links to the application that created them. See also object embedding, object linking.
object management workbench (OMW)	The change management system that is used for EnterpriseOne development.

object-based technology (OBT)	A technology that supports some of the main principles of object-oriented technology: Classes. Polymorphism.I Inheritance. Encapsulation.
object-oriented technology (OOT)	Brings software development past procedural programming into a world of re-usable programming that simplifies development of applications. Object orientation is based on the following principles: Classes. Polymorphism.I Inheritance. Encapsulation.
offsetting account	An account that reduces the amount of another account to provide a net balance. For example, a credit of 200 to a cash account might have an offsetting entry of 200 to an A/P Trade (liability) account.
open database connectivity (ODBC)	Defines a standard interface for different technologies to process data between applications and different data sources. The ODBC interface comprises set of function calls, methods of connectivity, and representation of data types that define access to data sources.
open systems interconnection (OSI)	The OSI model was developed by the International Standards Organization (ISO) in the early 1980s. It defines protocols and standards for the interconnection of computers and network equipment.
order detail line	A part of an order that contains transaction information about a service or item being purchased or sold, such as quantity, cost, price, and so on.
order hold	A flag that stops the processing of an order because it has exceeded the credit or budget limit, or has another problem.
order-based pricing	Pricing strategy that grants reductions in price to a customer. It is based upon the contents and relative size (volume or value) of the order as a whole.
outbound document	A document that is sent to a trading partner using EDI. This term is also referred to as an outbound transaction.

outturn	The quantity of oil that is actually received into a buyer's storage tanks when a vessel is unloaded. For various reasons (vaporization, clingage to vessel tank walls, and so on), the amount of a product pumped into shore tankage at unloading is often less than the quantity originally loaded onto the vessel, as certified by the Bill of Lading. Under a delivered or CIF outturn transaction, the buyer pays only for the barrels actually "turned out" by the vessel into storage. When a buyer is paying CIF Bill of Lading figures, a loss of 0.5% of total cargo volume is considered normal. Losses in excess of 0.5%, however, are either chargeable to the seller or are covered by specialized insurance that covers partial, as well as total, loss of the cargo.
overhead	In the distillation process, that portion of the charge that leaves the top of the distillation column as vapor. This definition is strictly as it relates to ECS.
override conversion method	A method of calculating exchange rates that is set up between two specific currencies. For those specific currencies, this method overrides the conversion method in General Accounting Constants and does not allow inverse rates to be used when calculating currency amounts.
package / package build	A collection of software that is grouped into a single entity for modular installation. EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where the installation program can find them on the deployment server. It is a point-in-time "snapshot" of the central objects on the deployment server.
package location	The directory structure location for the package and its set of replicated objects. This location is usually \\deployment server\release\path_code\package\ package name. The replicated objects for the package are placed in the subdirectories under this path. This location is also where the package is built or stored.
package workbench	During the Installation Workbench process, Package Workbench transfers the package information tables from the Planner data source to the System - release number data source. It also updates the Package Plan detail record to reflect completion.
packaged products	Products that, by their nature, must be delivered to the customer in containers which are suitable for discrete consumption or resale.
pane/panel	A resizable subarea of a window that contains options, components, or other related information.
paper clip	An icon that is used to indicate that a media object is attached to a form or record.
parent/child form	A type of form that presents parent/child relationships in an application on one form: The left portion of the form presents a tree view that displays a visual representation of a parent/child relationship. The right portion of the form displays a detail area in browse mode. The detail area displays the records for the child item in the tree. The parent/child form supports drag and drop functionality.

parent/child relationship	See parent/component relationship.
parent/component relationship	1. In Capital Asset Management, the hierarchical relationship of a parent piece of equipment to its components. For example, a manufacturing line could be a parent and the machinery on the line could be components of the line. In addition, each piece of machinery could be a parent of still more components. 2. In Product Data Management, a hierarchical relationship of the components and subassemblies of a parent item to that parent item. For example, an automobile is a parent item; its components and subassemblies include: engine, frame, seats, and windows. Sometimes referred to as parent/child relationship.
partita IVA	In Italy, a company fiscal identification number.
pass-through	A process where data is accepted from a source and forwarded directly to a target without the system or application performing any data conversion, validation, and so on.
pay on consumption	The method of postponing financial liability for component materials until you issue that material to its consuming work order or rate schedule.
payment group	A system-generated group of payments with similar information, such as a bank account. The system processes all of the payments in a payment group at the same time.
PeopleSoft database	See JDEBASE Database Middleware.
performance tuning	The adjustments that are made for a more efficient, reliable, and fast program.
persistent object	An object that continues to exist and retains its data beyond the duration of the process that creates it.
pervasive device	A type of intelligent and portable device that provides a user with the ability to receive and gather information anytime, from anywhere.
planning family	A means of grouping end items that have similarity of design or manufacture.
plug-in	A small program that plugs into a larger application to provide added functionality or enhance the main application.
polymorphism	A principle of object-oriented technology in which a single mnemonic name can be used to perform similar operations on software objects of different types.
portal	A Web site or service that is a starting point and frequent gateway to a broad array of on-line resources and services.
Postfinance	A subsidiary of the Swiss postal service. Postfinance provides some banking services.

potency	Identifies the percent of an item in a given solution. For example, you can use an 80% potent solution in a work order that calls for 100% potent solution, but you would use 25% more, in terms of quantity, to meet the requirement ($100 / 80 = 1.25$).
preference profile	The ability to define default values for specified fields for a user defined hierarchy of items, item groups, customers, and customer groups. In Quality Management setup, this method links test and specification testing criteria to specific items, item groups, customers, or customer groups.
preflush	A work order inventory technique in which you deduct (relieve) materials from inventory when the parts list is attached to the work order or rate schedule.
preventive maintenance cycle	The sequence of events that make up a preventive maintenance task, from its definition to its completion. Because most preventive maintenance tasks are commonly performed at scheduled intervals, parts of the preventive maintenance cycle repeat, based on those intervals.
preventive maintenance schedule	The combination of service types that apply to a specific piece of equipment, as well as the intervals at which each service type is scheduled to be performed.
primary service type	A service type to which you can link related service types. For example, for a particular piece of equipment, you might set up a primary service type for a 1000-hour inspection and a linked service type for a 500-hour inspection. The 1000-hour inspection includes all of the tasks performed at 500 hours. When a primary service type is scheduled to be performed, the system schedules the linked service type. See also linked service type.
pristine environment	An EnterpriseOne environment that is used to test unaltered objects with PeopleSoft demonstration data or for training classes. You must have this environment so you can compare pristine objects that you modify.
processing option	A data structure that allows users to supply parameters that regulate the execution of a batch program or report.
product data management (PDM)	In PeopleSoft EnterpriseOne software, the system that enables a business to organize and maintain information about each item which it manufactures. Features of this system, such as bills of material, work centers, and routings, define the relationships among parents and components, and how they can be combined to manufacture an item. PDM also provides data for other manufacturing systems including Manufacturing Accounting, Shop Floor Management, and Manufacturing and Distribution Planning.
product line	A group of products with similarity in manufacturing procedures, marketing characteristics, or specifications that allow them to be aggregated for planning; marketing; and, occasionally, costing.
product/process definition	A combination of bill of material (recipe, formula, or both) and routing (process list). Organized into tasks with a statement of required consumed resources and produced resources.
production environment	An EnterpriseOne environment in which users operate EnterpriseOne software.

program temporary fix (PTF)	A representation of changes to PeopleSoft software that your organization receives on magnetic tapes or diskettes.
project	[In EnterpriseOne] A virtual container for objects being developed in Object Management Workbench.
projected cost	The target expenditure in added value for material, labor, and so on, during manufacture. See also standard cost.
promotion path	The designated path for advancing objects or projects in a workflow.
protocollo	See registration number.
PST	Provincial sales tax. A tax that is assessed by individual provinces in Canada.
published table	Also called a “Master” table, this is the central copy to be replicated to other machines and resides on the “publisher” machine. The Data Replication Publisher Table (F98DRPUB) identifies all of the published tables and their associated publishers in the enterprise.
publisher	The server that is responsible for the published table. The Data Replication Publisher Table (F98DRPUB) identifies all of the published tables and their associated publishers in the enterprise.
pull replication	One of the EnterpriseOne methods for replicating data to individual workstations. Such machines are set up as pull subscribers that use EnterpriseOne’s data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the Data Replication Pending Change Notification table (F98DRPCN).
query by example (QBE)	Located at the top of a detail area, this area is used to search for data to display in the detail area.
rate scheduling	A method of scheduling product or manufacturing families, or both. Also a technique to determine run times and quantities of each item within the family to produce enough of each individual product to satisfy demand until the family can be scheduled again.
rate type	For currency exchange transactions, the rate type distinguishes different types of exchange rates. For example, you can use both period average and period-end rates, distinguishing them by rate type.
real-time	Pertaining to information processing that returns a result so rapidly that the interaction appears to be instantaneous.
receipt routing	A series of steps that is used to track and move items within the receipt process. The steps might include in-transit, dock, staging area, inspection, and stock.
referential integrity	Ensures that a parent record cannot be deleted from the database when a child record for exists.

regenerable	Source code for EnterpriseOne business functions can be regenerated from specifications (business function names). Regeneration occurs whenever an application is recompiled, either for a new platform or when new functionality is added.
register types and classes	In Italian VAT Summary Reporting, the classification of VAT transactions.
relationship	Links tables together and facilitates joining business views for use in an application or report. Relationships that are created are based on indexes.
relevé d'identité bancaire (RIB)	In France, the term that indicates the bank transit code, account number, and check digit that are used to validate the bank transit code and account number. The bank transit code consists of the bank code and agency code. The account number is alphanumeric and can be as many as 11 characters. PeopleSoft supplies a validation routine to ensure RIB key correctness.
remessa	In Brazil, the remit process for A/R.
render	To include external data in displayed content through a linking mechanism.
repasse	In Brazil, a discount of the ICMS tax for interstate transactions. It is the adjustment between the interstate and the intrastate ICMS tax rates.
replenishment point	The location on or near the production line where additional components or subassemblies are to be delivered.
replication server	A server that is responsible for replicating central objects to client machines.
report design aid (RDA)	The EnterpriseOne GUI tool for operating, modifying, and copying report batch applications.
repost	In Sales, the process of clearing all commitments from locations and restoring commitments, based on quantities from the Sales Order Detail table (F4211).
resident	Pertaining to computer programs or data while they remain on a particular storage device.
retorno	In Brazil, the receipt process for A/R.
RIB	See relevé d'identité bancaire.
ricevute bancarie (RiBa)	In Italy, the term for accounts receivable drafts.
riepilogo IVA	Summary VAT monthly report. In Italy, the term for the report that shows the total amount of VAT credit and debit.
ritenuta d'acconto	In Italy, the term for standard withholding tax.
rollback	[In database management] A feature or command that undoes changes in database transactions of one or more records.
rollup	See cost rollup.

row exit	[In EnterpriseOne] An application shortcut, available as a button on the Row Exit bar or as a menu selection, that allows users to open a form that is related to the highlighted grid record.
runtime	The period of time when a program or process is running.
SAD	The German name for a Swiss payment format that is accepted by Postfinance.
SAR	See software action request.
scalability	The ability of software, architecture, hardware, or a network to support software as it grows in size or resource requirements.
scripts	A collection of SQL statements that perform a specific task.
scrub	To remove unnecessary or unwanted characters from a string.
search/select	A type of form that is used to search for a value and return it to the calling field.
selection	Found on PeopleSoft menus, selections represent functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.
serialize	To convert a software object into a stream of bytes to store on a disk or transfer across a network.
server map	The server view of the object configuration mapping.
server workbench	During the Installation Workbench process, Server Workbench copies the server configuration files from the Planner data source to the System release number data source. It also updates the Server Plan detail record to reflect completion.
service interval	The frequency at which a service type is to be performed. Service intervals can be based on dates, periods, or statistical units that are user defined. Examples of statistical units are hours, miles, and fuel consumption.
service type	An individual preventive maintenance task or procedure, such as an inspection, lubrication, or overhaul. Service types can apply to a specific piece of equipment or to a class of equipment. You can specify that service types come due based on a predetermined service interval, or whenever the task that is represented by the service type becomes necessary.
servlet	A [small] program that extends the functionality of a Web server by generating dynamic content and interacting with Web clients by using a request-response paradigm.
share path	The network node under which one or more servers or objects reside.
shop floor management	A system that uses data from multiple system codes to help develop, execute, and manage work orders and rate schedules in the enterprise.
silent mode	A method for installing or running a program that does not require any user intervention.

silent post	A type of post that occurs in the background without the knowledge of the user.
simulated cost	After a cost rollup, the cost of an item, operation, or process according to the current cost scenario. This cost can be finalized by running the frozen update program. You can create simulated costs for a number of cost methods—for example, standard, future, and simulated current costs. See also cost rollup.
single-byte character set (SBCS)	An encoding scheme in which each alphabetic character is represented by one byte. Most Western languages, such as English, can be represented by using a single-byte character set.
single-level tracking	Finding all immediate parents where a specific lot has been used (consumed).
single-voyage (spot) charter	An agreement for a single voyage between two ports. The payment is made on the basis of tons of product delivered. The owner of the vessel is responsible for all expenses.
slimer	A script that changes data in a table directly without going through a regular database interface.
smart field	A data dictionary item with an attached business function for use in the Report Design Aid application.
SOC	The Italian term for a Swiss payment format that is accepted by Postfinance.
soft commitment	The number of items that is reserved for sales orders or work orders in the primary units of measure.
soft error	An error from which an operating system or program is able to recover.
software action request (SAR)	An entry in the AS/400 database that is used for requesting modifications to PeopleSoft software.
SOG	The French term for a Swiss payment format that is accepted by Postfinance.
source directory	The path code to the business function source files belonging to the shared library that is created on the enterprise server.
special period/year	The date that determines the source balances for an allocation.
specification merge	The Specification merge is comprised of three merges: - Object Librarian merge (via the Object Management Workbench). - Versions List merge. - Central Objects merge. The merges blend customer modifications with data that accompanies a new release.
specification table merge workbench	During the Installation Workbench process, Specification Table Merge Workbench runs the batch applications that update the specification tables.

specifications	A complete description of an EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.
spot charter	See single-voyage charter.
spot rates	An exchange rate that is entered at the transaction level. Spot rates are not used on transactions between two EMU member currencies because exchange rates are irrevocably fixed to the euro.
stamp tax	In Japan, a tax that is imposed on drafts payable, receipts over 30000 Japanese yen, and all contracts. The party that issues any of the above documents is responsible for this tax.
standalone	Operating or capable of operating independently of certain other components of a computer system.
standard cost	The expected, or target cost of an item, operation, or process. Standard costs represent only one cost method in the Product Costing system. You can also calculate, for example, future costs or current costs. However, the Manufacturing Accounting system uses only standard frozen costs.
standard costing	A costing method that uses cost units that are determined before production. For management control purposes, the system compares standard costs to actual costs and computes variances.
subprocess	A process that is triggered by and is part of a larger process, and that generally consists of activities.
subscriber table	The Subscriber table (F98DRSUB), which is stored on the Publisher Server with the Data Replication Publisher table (F98DRPUB), that identifies all of the subscriber machines for each published table.
summary	The presentation of data or information in a cumulative or totaled manner in which most of the details have been removed. Many systems offer forms and reports that summarize information which is stored in certain tables. Contrast with detail.
super backflush	To create backflush transactions for material, labor, or both, against a work order at predefined pay points in the routing. By doing so, you can relieve inventory and account for labor amounts at strategic points throughout the manufacturing process.
supersession	Specification that a new product is replacing an active product on a specified effective date.
supplemental data	Additional types of data for customers and suppliers. You can enter supplemental data for information such as notes, comments, plans, or other information that you want in a customer or supplier record. The system maintains this data in generic databases, separate from the standard master tables (Customer Master, Supplier Master, and Address Book Master).

supplying location	The location from which inventory is transferred once quantities of the item on the production line have been depleted. In kanban processing, the supplying location is the inventory location from which materials are transferred to the consuming location when the containers are replenished.
system code	A numeric or alphanumeric designation that identifies a specific system in EnterpriseOne software.
system function	[In EnterpriseOne] A named set of pre-packaged, re-usable instructions that can be called from event rules.
table access management (TAM)	The EnterpriseOne component that handles the storage and retrieval of user defined data. TAM stores information such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions.
table conversion workbench	During the Installation Workbench process, Table Conversion Workbench runs the table conversions that change the technical and application tables to the format for the new release of EnterpriseOne. It also updates the Table Conversions and Controls detail records to reflect completion.
table design aid (TDA)	An EnterpriseOne GUI tool for creating, modifying, copying, and printing database tables.
table event rules	Use table event rules to attach database triggers (or programs) that automatically run whenever an action occurs against the table. An action against a table is referred to as an event. When you create an EnterpriseOne database trigger, you must first determine which event will activate the trigger. Then, use Event Rules Design to create the trigger. Although EnterpriseOne allows event rules to be attached to application events, this functionality is application-specific. Table event rules provide embedded logic at the table level.
table handle	A pointer into a table that indicates a particular row.
table space	[In relational database management systems] An abstract collection of containers in which database objects are stored.
task	[In Solution Explorer and EnterpriseOne Menu] A user defined object that can initiate an activity, process, or procedure.
task view	A group of tasks in Solution Explorer or EnterpriseOne Menu that are arranged in a tree structure.
termo de abertura	In Brazil, opening terms for the transaction journal.
termo de encerramento	In Brazil, closing terms for the transaction journal.
three-tier processing	The task of entering, reviewing, approving, and posting batches of transactions.
three-way voucher match	The process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records, the purchase order, and the invoice to create vouchers.

threshold percentage	In Capital Asset Management, the percentage of a service interval that you define as the trigger for maintenance to be scheduled. For example, you might set up a service type to be scheduled every 100 hours with a threshold percentage of 90 percent. When the equipment accumulates 90 hours, the system schedules the maintenance.
throughput agreement	A service agreement in which a business partner agrees to store and manage product for another business partner for a specified time period. The second partner actually owns the stock that is stored in the first partner's depot, although the first partner monitors the stock level; suggests replenishments; and unloads, stores, and delivers product to the partner or its customers. The first partner charges a fee for storing and managing the product.
throughput reconciliation	Reconcile confirmed sales figures in a given period with the measured throughput, based on the meter readings. This process is designed to catch discrepancies that are due to transactions not being entered, theft, faulty meters, or some combination of these factors. This reconciliation is the first stage. See also operational reconciliation.
token	[In Object Management Workbench] A flag that is associated with each object which indicates whether you can check out the object.
tolerance range	The amount by which the taxes that you enter manually can vary from the tax that is calculated by the system.
TP monitor	Transaction Processing monitor. A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and can include programs that validate data and format terminal screens.
tracing	The act of researching a lot by going backward, to discover its origin.
tracking	The act of researching a lot by going forward, to discover where it is used.
transaction set	An electronic business transaction (EDI Standard document) composed of segments.
transclude	To include the external data in the displayed content through a linking mechanism.
transfer order	An order that is used to ship inventory between branch/plants within your company and to maintain an accurate on-hand inventory amount. An interbranch transfer order creates a purchase order for the shipping location and a sales order for the receiving location.
translation adjustment account	An optional G/L account used in currency balance restatement to record the total adjustments at a company level.
translator software	The software that converts data from an application table format to an EDI Standard Format, and from EDI Standard Format to application table format. The data is exchanged in an EDI Standard, such as ANSI ASC X12, EDIFACT, UCS, or WINS.

tree structure	A type of graphical user interface that displays objects in a hierarchy.
trigger	Allows you to attach default processing to a data item in the data dictionary. When that data item is used on an application or report, the trigger is invoked by an event which is associated with the data item. EnterpriseOne also has three visual assist triggers: Calculator. Calendar. Search form.
two-way voucher match	The process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information.
universal batch engine (UBE)	[In EnterpriseOne] A type of application that runs a noninteractive process.
unnormalized	Data that is a random collection of data elements with repeating record groups scattered throughout. Also see Normalized.
user overrides merge	The User Overrides merge adds new user override records into a customer's user override table.
user-defined code (UDC)	A value that a user has assigned as being a valid entry for a given or specific field.
utility	A small program that provides an addition to the capabilities which are provided by an operating system.
variable numerator allocations	A procedure that allocates or distributes expenses, budgets, adjustments, and so on, among business units, based on a variable.
variable quantity	A term that indicates the bill of material relationship between a parent item and its components or ingredients. When a bill of material component has a variable quantity relationship to its parent, the amount of the component changes when the software calculates parts list requirements for different work order quantities. Contrast with fixed quantity.
variance	1. In Product Costing and Manufacturing Accounting, the difference between the frozen standard cost, the current cost, the planned cost, and the actual cost. For example, the difference between the frozen standard cost and the current cost is an engineering variance. Frozen standard costs come from the Cost Components table, and the current costs are calculated by using the current bill of material, routing, and overhead rates. 2. In Capital Asset Management, the difference between revenue that is generated by a piece of equipment and costs that are incurred by the equipment.
versions list merge	The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release as well as their processing options data.
VESR	Verfahren Einzahlungsschein mit Referenznummer. The processing of an ESR pay slip with reference line through accounts receivable and accounts payable.

visual assist	Forms that can be invoked from a control to assist the user in determining what data belongs in the control.
voucher logging	The process of entering vouchers without distributing amounts to specific G/L accounts. The system initially distributes the total amount of each voucher to a G/L suspense account, where it is held until you redistribute it to the correct G/L account.
wareki date format	In Japan, a calendar format, such as Showa or Heisei. When a new emperor begins to reign, the government chooses the title of the date format and the year starts over at one. For instance, January 1, 1998, is equal to Heisei 10, January 1st.
wash down	A minor cleanup between similar product runs. Sometimes used in reference to the sanitation process of a food plant.
wchar_t	An internal type of a wide character. Used for writing portable programs for international markets.
web server	A server that sends information as requested by a browser and uses the TCP/IP set of protocols.
work order life cycle	In Capital Asset Management, the sequence of events through which a work order must pass to accurately communicate the progress of the maintenance tasks that it represents.
workfile	A system-generated file that is used for temporary data processing.
workflow	According to the Workflow Management Coalition, workflow means “the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.”
workgroup server	A network server usually containing subsets of data that are replicated from a master network server.
WorldSoftware architecture	The broad spectrum of application design and programming technology that PeopleSoft uses to achieve uniformity, consistency, and complete integration throughout its software.
write payment	A step in processing payments. Writing payments includes printing checks, drafts, and creating a bank tape table.
write-off	A method for getting rid of inconsequential differences between amounts. For example, you can apply a receipt to an invoice and write off the difference. You can write off both overpayments and underpayments.

Z file	For store and forward (network disconnected) user, EnterpriseOne store-and-forward applications perform edits on static data and other critical information that must be valid to process an order. After the initial edits are complete, EnterpriseOne stores the transactions in work tables on the workstation. These work table are called Z files. When a network connection is established, Z files are uploaded to the enterprise server; and the transactions are edited again by a master business function. The master business function then updates the records in your transaction files.
z-process	A process that converts inbound data from an external system into an EnterpriseOne software table or converts outbound data into an interface table for an external system to access.
zusammenfassende melding	In Germany, the term for the EU Sales Listing.

Index

A

- Auto commit
 - Java connector, 97
- Auto transaction
 - dynamic Java connector, 80

B

- BHVRCOM
 - Java connector, 97
- BSFN cache
 - dynamic Java connector, 80
- BSFNMethod
 - dynamic Java connector, 69
- BSFNParameter
 - dynamic Java connector, 69
- BSFNSpecSource
 - dynamic Java connector, 69
- Business function
 - using BHVRCOM, 97
- Business function call
 - dynamic Java connector, 78
- business function metadata
 - dynamic Java connector, 68
- Business function wrapper
 - generate using Visual Basic, 16
- Business functions
 - validating spec metadata, 73

C

- Cache
 - dynamic Java connector, 80
- CheckVer
 - Java connector, 89
 - migrating from previous release, 89
- CheckVer Java connector, 90
- CLASSPATH settings
 - resource adapter, 119
- COM
 - connector, 8
 - events support, 48
 - IJDETimezone interface, 30
 - installing on non-PeopleSoft EnterpriseOne client, 19
 - objects, 6
 - overview, 6
 - PeopleSoft EnterpriseOne server, 8

- reliability, 47
- server deployment, 19
- subscribe to events, 51
- tracing and logging, 46
- troubleshooting, 47
- using BHVRCOM, 31
- wrapper version checker
 - running, 29
 - using, 29

- COM Components
 - generating, 9
- COM connector
 - create using C++, 16
 - inbound XML request, 32
 - OCM support, 45
- COM transactions
 - auto commit, 32
 - calling prepare and commit, 32
 - manual commit, 32
- COM+
 - registering components, 48
 - registering EventSink, 53
- Common client interface
 - resource adapter, 121
- Compton-phase commit transaction, 33
- configurable properties
 - resource adapter, 120
- Configuring dynamic Java connector
 - non-PeopleSoft EnterpriseOne machine, 77
- Configuring Java connector
 - non-PeopleSoft EnterpriseOne machine, 90
- Connectors
 - choosing, 4

D

- data
 - resource adapter, 124
- DCOM
 - client environment
 - set up, 26
 - identity
 - set up, 24
 - server environment
 - set up, 21
 - server security
 - set up, 21
- DCOM server
 - setup, 21

- Dynamic java connector
 - logging, 83
- Dynamic Java connector
 - and PeopleSoft EnterpriseOne, 68
 - BSFN cache, 80
 - BSFNMethod, 69
 - BSFNParameter, 69
 - BSFNSpecSource, 69
 - exception handling, 84
 - inbound XML request, 82
 - OCM support, 81
 - overview, 68
 - SpecDictionary, 71
 - transactions, 80
 - user session management, 81
- Dynamic Java Connector
 - design considerations, 68
 - run time considerations, 78
- Dynamic Java connector implementing an
 - interface for events, 110
- Dynamic Java connector installing components
 - non-PeopleSoft EnterpriseOne machine, 77
- Dynamic Java connector subscribing to events, 110
- Dynamic mode configuration
 - Java connector, 89

E

- Error handling
 - dynamic Java connector, 84
 - Java connector, 100
- Events
 - COM, 48
 - compile and run Java client, 113
 - implement an interface, 110
 - setting up Java client, 110
 - subscribe to an event, 110
- Example code
 - BizTalk integration, 62
 - ClientPrj for distributed transaction, 44
 - COMConnector.dll, 55
 - common client interface, 121
 - create COM+ component, 52
 - Java connector exception handling, 105
 - MTStest for distributed transaction, 43
 - sales order entry transactional client, 41
 - sales order entry transactional object, 39
 - subscribe to an event, 111
 - subscriber, 58
- Exception handling
 - dynamic Java connector, 84
 - exception details, 101

- fatal exception, 100
- Java connector, 100
- recoverable exception, 101
- reject, 101
- resource adapter, 125

G

- GenCom
 - options, 13
 - output, 16
 - running, 13
 - syntax, 13
- GenCOM
 - client environment
 - include directories, 11
 - lib directories, 11
 - MSDev directories, 12
 - paths, 12
 - environment setup, 10
 - flow, 9
 - installation, 46
 - ProgID, 15
 - using, 13
- GenJava
 - options, 92
 - overview, 91
 - syntax, 92
 - using output, 94
- GenJAVA
 - client environment
 - set up, 90
 - overview, 88
- GenJava CheckVer, 90

I

- iJDEScript commands, 139
 - Build, 139
 - Call, 140
 - Define, 140
 - Define!, 140
 - Exit, 141
 - Help, 141
 - Import, 141
 - Importlib, 142
 - Interface, 142
 - Library, 142
 - Login, 143
 - Logout, 143
 - Opt, 143
 - Rename, 143
 - Say, 144

- Sub, 144
- System, 145
- IJDETimeZone interface
 - COM, 30
- ImageBSFNInteractionSpecImpl, 123
- Installing dynamic Java connector
 - non-PeopleSoft EnterpriseOne machine, 77
- Installing Java connector
 - non-PeopleSoft EnterpriseOne machine, 90

J

- Java
 - overview, 66
- Java connector
 - user session management, 99
- Java connector
 - and PeopleSoft EnterpriseOne, 86
 - BHVRCOM structure, 97
 - CheckVer, 89
 - compile Java client, 113
 - exception handling, 100
 - implementing an interface for events, 110
 - inbound XML request, 100
 - OCM support, 98
 - overview, 86
 - run Java client, 113
 - setting up to process events, 110
 - subscribing to events, 110
 - transaction, 97
 - versioning
 - static and dynamic modes, 89
- Java Connector
 - design considerations, 88
 - run time considerations, 91
- Java Connector
 - versioning, 88
- Java connector architecture resource adapter, 113. *See Resource adapter*
 - overview, 113
- Java connector exception handling
 - exception details, 101
 - fatal errors, 100
 - recoverable errors, 101
 - reject, 101
- Java connector installing components
 - non-PeopleSoft EnterpriseOne machine, 90
- Java dynamic connector
 - setting up to process events, 110
- Java dynamic connector compile Java client, 113
- Java dynamic connector run Java client, 113
- Java Naming and Directory Interface. *See* JNDI
- Java wrapper version checker, 89

- jdeinterop settings
 - resource adapter, 119
- jdeinterop.ini file
 - section settings
 - [CACHE], 130
 - [DEBUG], 131
 - [EVENTS], 133
 - [INTEROP], 132
 - [JDBj-BOOTSTRAP DATA SOURCE], 134
 - [JDBj-BOOTSTRAP SESSION], 133
 - [JDBj-CONNECTION POOL], 135
 - [JDBj-JDBC DRIVERS], 135
 - [JDBj-ORACLE], 136
 - [JDENET], 131
 - [OCM], 130
 - [SECURITY], 131
 - [SERVER], 131
- jdeinterop.ini settings
 - overview, 130
- jdeinterop.ini settings for events, 51
- jdeLog.properties
 - settings, 137
- jdelog.properties settings
 - resource adapter, 119
- JNDI settings
 - resource adapter, 120

L

- Logging
 - COM events, 52
 - dynamic Java connector, 83
 - resource adapter, 125

M

- Manual commit
 - Java connector, 97
- Manual transaction
 - dynamic Java connector, 80
- Messages
 - dynamic Java connector, 83

O

- OCM support
 - Dynamic Java connector, 81
 - Java connector, 98
- Outbound events
 - dynamic Java connector, 108
 - Java connector, 108
- Overview
 - COM, 6

- connectors, 4
- dynamic Java connector, 68
- Java, 66
- Java connector, 86
- Java connector architecture resource adapter, 113
- OWBSFNInteractionSpecImpl, 123

P

- PeopleSoft EnterpriseOne and COM, 6
- PeopleSoft EnterpriseOne and dynamic Java connector, 68
- PeopleSoft EnterpriseOne and Java, 66
- PeopleSoft EnterpriseOne and Java connector, 86

R

- Registering COM+ components, 48
- resource adapter
 - signon types, 122
- Resource adapter, 113
 - assembly, 117
 - classpath settings, 119
 - common client interface, 121
 - components, 117
 - configurable properties, 120
 - configuration, 118
 - deployment, 118
 - exceptions, 125
 - features, 114
 - input and output data, 124
 - JCA 1.0 specification, 114
 - jdeinterop settings, 119
 - jdelog.properties settings, 119
 - JNDI settings, 120
 - samples, 125
 - security permissions, 118
 - subclasses, 123
 - troubleshooting, 128
- Resource adapter interface
 - subclasses, 123
- Running CheckVer, 29
- Running GenJava, 92

S

- Sample applications
 - compiling, 85
 - running, 85
 - shipped, 84
- Signon types
 - resource adapter, 122
- Spec image
 - generate, 74, 75
 - synchronize, 77
 - validate, 76
- SpecDictionary
 - dynamic Java connector, 71
- Static mode configuration
 - Java connector, 89
- Subscribing to events for COM, 51

T

- Transaction
 - Java connector, 97
- Transactions
 - dynamic Java connector, 80
- Troubleshooting
 - COM, 47
 - resource adapter, 128

U

- User session management
 - dynamic Java connector, 81
 - Java connector, 99
- Using BHVRCOM via COM, 31
- Using GenJava output, 94

V

- Versioning
 - Java connector, 88

X

- XML request
 - using COM connector, 32
 - using Dynamic Java connector, 82
 - using Java connector, 100