

# PeopleSoft®

---

## Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Component Interfaces

---

June 2004

## Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Component Interfaces SKU PT845CPI-B 0604

Copyright © 1988-2004 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

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. The following open source software may be used in PeopleSoft products and the following disclaimers are provided.

#### Apache Software Foundation

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.

#### OpenSSL

Copyright (c) 1998-2003 The OpenSSL Project. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "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 OpenSSL PROJECT 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.

#### SSLLeay

Copyright (c) 1995-1998 Eric Young. All rights reserved.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS 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 AUTHOR OR 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.

#### Loki Library

Copyright (c) 2001 by Andrei Alexandrescu. This code accompanies the book:

Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright (c) 2001. Addison-Wesley. Permission to use, copy, modify, distribute and sell this software for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

# Contents

## General Preface

- About This PeopleBook .....ix**
- PeopleSoft Application Prerequisites.....ix
- PeopleSoft Application Fundamentals.....ix
- Related Documentation.....x
  - Obtaining Documentation Updates.....x
  - Ordering Printed Documentation.....x
- Typographical Conventions and Visual Cues.....xi
  - Typographical Conventions.....xi
  - Visual Cues.....xii
  - Country, Region, and Industry Identifiers.....xii
  - Currency Codes.....xiii
- Comments and Suggestions.....xiii
- Common Elements in These PeopleBooks .....xiii

## Preface

- PeopleSoft Component Interfaces Preface.....xv**
- PeopleSoft Component Interfaces.....xv

## Chapter 1

- Getting Started with PeopleSoft Component Interfaces.....1**
- PeopleSoft Component Interfaces Technology Overview.....1
- Implementing PeopleSoft Component Interfaces.....1
  - Implementing the Excel to Component Interfaces Utility.....2

## Chapter 2

- Introducing Component Interfaces.....3**
- Understanding Component Interfaces.....3
- Component Interface Architecture.....3
- Component Interface Attributes.....4
  - Name.....4
  - Keys.....4

Properties.....4  
 Collections.....4  
 Methods.....5  
 Component Interface Definitions and Views.....5

**Chapter 3**

**Developing the Component Interface.....9**

Creating the Component Interface Definition.....9  
 Creating a New Component Interface.....9  
 Criteria for Defaulting Properties.....11  
 Naming the Component Interface Definition.....11  
 Associating a Component Interface with a Menu.....12  
 Determining Which Fields to Expose.....13  
 Using Keys.....13  
 Understanding Keys.....13  
 Adding and Deleting Keys.....15  
 Setting Properties.....15  
 Standard Properties.....15  
 Creating User-Defined Properties.....27  
 Deleting User-Defined Properties.....28  
 Renaming User-Defined Properties.....28  
 Creating Reference Properties.....29  
 Making Properties Read-Only.....31  
 Working with Collections.....31  
 Working with Methods.....31  
 Working with Session Functions and Methods.....32  
 Standard Methods.....33  
 Collection Methods.....37  
 Enabling and Disabling Standard Methods.....41  
 Creating User-Defined Methods.....41  
 Validating the Component Interface.....43  
 Setting Component Interface Security.....43  
 Testing the Component Interface.....44  
 Searching for a Component Interface to Test.....45  
 Testing the Component Interface.....47  
 Determining ItemByKeys Parameters.....50  
 Understanding Synchronization.....51  
 Writing a Component Interface Program.....52  
 Runtime Considerations.....53

General Considerations.....53  
 Scope Conflicts.....53  
 Interactive Mode.....54

**Chapter 4**

**Programming Component Interfaces in Java.....55**  
 Building APIs in Java.....55  
 Setting Up the Java Environment.....56  
 Generating a Java Runtime Code Template.....56  
 Understanding the Java Template.....58

**Chapter 5**

**Programming Component Interfaces in C++.....61**  
 Building APIs for C++.....61  
 Setting Up the C++ Environment.....61  
     Setting Up a Client Machine to Access a C++ API.....62  
     Configuring a Compiler for the C++ Project.....62  
 Generating a C++ Runtime Code Template.....63  
 Understanding the C++ Template.....64

**Chapter 6**

**Programming Component Interfaces in COM.....69**  
 Building APIs for COM.....69  
 Setting Up the COM Environment.....70  
 Generating a Visual Basic Runtime Code Template.....71  
 Understanding the Visual Basic Template.....72

**Chapter 7**

**Using the Component Interface Software Development Kit.....75**  
 Understanding the Component Interface SDK.....75  
     Component Interface SDK Samples.....75  
 Setting SDK Prerequisites.....75  
 Using the SDK\_BUS\_EXPENSES Test Page.....76  
 Testing the SDK\_BUS\_EXP .....76  
 Using the SDK Java Sample.....77  
     Running the Java Sample.....77

Understanding the Java Sample Code.....	77
Using the SDK C++ Sample.....	79
Building the C++ Sample.....	80
Running the SDK C++ Sample.....	80
Understanding the C++ Sample Code.....	80
Using the SDK COM Excel Sample.....	82
Running the COM Excel Sample.....	82
Understanding the COM Excel Sample Code.....	83
Using the SDK COM ASP Sample.....	84
Running the COM ASP Sample.....	84
Understanding the COM ASP Sample Code.....	85

## Chapter 8

<b>Programming Component Interfaces in PeopleCode.....</b>	<b>91</b>
Understanding PeopleCode Behavior and Limitations.....	91
PeopleCode Event and Function Behavior.....	91
CopyRowset Language Considerations.....	92
Limitations of Client-Only PeopleCode.....	92
Generating a PeopleCode Template.....	93
Understanding the PeopleCode Template.....	94

## Chapter 9

<b>Using the Excel to Component Interface Utility.....</b>	<b>97</b>
Understanding the Excel to Component Interface Utility .....	97
Building a Component Interface for the Excel to Component Interface Utility.....	98
Getting Started with the Excel to Component Interface Utility.....	100
Viewing the Coversheet.....	101
Setting Up Connection Information.....	101
Entering Connection Information.....	101
Connecting to the Database to Create a Template and Submit Data.....	104
Creating a Template.....	105
Understanding the Template Actions Toolbar.....	106
Entering Data into the Template.....	108
Template Wrapping.....	109
Entering Data on the Data Input Sheet.....	109
Using the Data Input Sheet.....	109
Viewing the Staged Data.....	110
Correcting and Resubmitting Data.....	112

Creating a SOAP/XML Request.....113

    Request Format.....113

    Sample Create Request.....113

    Sample Get Request.....113

    Sample Update Request.....114

    Sample Updatedata Request.....114

Sending the Request.....115

Receiving a Response.....115

    Viewing a Response if a Row Already Exists.....115

    Viewing a Sample Get Request and Response.....116

Diagnosing and Resolving Errors.....117

    Viewing Log Files.....117

    Resolving Error Messages for Client Environments.....117

**Chapter 10**

**Using WSDL Binding for Component Interfaces.....119**

Understanding WSDL and Component Interfaces.....119

Setting Up Integration Broker.....119

    Ensuring That the SOAPTOCI Message Is Active.....120

    Verifying the Message Channel.....120

    Modifying the Local Gateway.....120

Creating the Third-Party Message Node.....121

Adding Security to PeopleSoft Objects.....122

Generating WSDL for a Component Interface.....123

Sending a SOAP Message to a PeopleSoft Application.....125

Understanding SOAP Considerations.....125

Testing SOAP to Component Interface by Using Send Master.....126

Viewing an Example of a Find SOAP Request.....127

**Appendix A**

**ISO Country and Currency Codes.....129**

ISO Country Codes.....129

ISO Currency Codes.....138

**Glossary of PeopleSoft Terms.....149**

**Index .....165**

# About This PeopleBook

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

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Related documentation.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

---

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

---

---

## PeopleSoft Application Prerequisites

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

See *Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications*.

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

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

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

---

## PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft database. However, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals PeopleBook. Each PeopleSoft product line has its own version of this documentation.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across a product line. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of this central PeopleBook. It is the starting point for fundamentals, such as setting up control tables and administering security.

---

## Related Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

### Obtaining Documentation Updates

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

---

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

---

#### See Also

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

### Ordering Printed Documentation

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

- Web
- Telephone
- Email

#### Web

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

#### Telephone

Contact MMA Partners at 877 588 2525.

#### Email

Send email to MMA Partners at [peoplesoftpress@mmapartner.com](mailto:peoplesoftpress@mmapartner.com).

#### See Also

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

## Typographical Conventions and Visual Cues

This section discusses:

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

### Typographical Conventions

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

Typographical Convention or Visual Cue	Description
<b>Bold</b>	Indicates PeopleCode function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
<i>Italics</i>	Indicates field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply.  We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter <i>O</i> .
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
Monospace font	Indicates a PeopleCode program or other code example.
“ ” (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe ( ).

Typographical Convention or Visual Cue	Description
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	<p>When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object.</p> <p>Ampersands also precede all PeopleCode variables.</p>

## Visual Cues

PeopleBooks contain the following visual cues.

### Notes

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

---

**Note.** Example of a note.

---

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

---

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

---

### Warnings

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

---

**Warning!** Example of a warning.

---

### Cross-References

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

## Country, Region, and Industry Identifiers

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

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

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

### Country Identifiers

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

See *About These PeopleBooks*, “ISO Country and Currency Codes,” ISO Country Codes.

## Region Identifiers

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

- Asia Pacific
- Europe
- Latin America
- North America

## Industry Identifiers

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

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

## Currency Codes

Monetary amounts are identified by the ISO currency code.

See Appendix A, “ISO Country and Currency Codes,” ISO Currency Codes.

## Comments and Suggestions

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

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

Or send email comments to [doc@peoplesoft.com](mailto:doc@peoplesoft.com).

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

## Common Elements in These PeopleBooks

<b>As of Date</b>	The last date for which a report or process includes data.
<b>Business Unit</b>	An ID that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization.
<b>Description</b>	Enter up to 30 characters of text.
<b>Effective Date</b>	The date on which a table row becomes effective; the date that an action begins. For example, to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when

you can view and change the information. Pages or panels and batch processes that use the information use the current row.

**Once, Always, and Don't Run**

Select Once to run the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to Don't Run.

Select Always to run the request every time the batch process runs.

Select Don't Run to ignore the request when the batch process runs.

**Report Manager**

Click to access the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list).

**Process Monitor**

Click to access the Process List page, where you can view the status of submitted process requests.

**Run**

Click to access the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format.

**Request ID**

An ID that represents a set of selection criteria for a report or process.

**User ID**

An ID that represents the person who generates a transaction.

**SetID**

An ID that represents a set of control table information, or TableSets. TableSets enable you to share control table information and processing options among business units. The goal is to minimize redundant data and system maintenance tasks. When you assign a setID to a record group in a business unit, you indicate that all of the tables in the record group are shared between that business unit and any other business unit that also assigns that setID to that record group. For example, you can define a group of common job codes that are shared between several business units. Each business unit that shares the job codes is assigned the same setID for that record group.

**Short Description**

Enter up to 15 characters of text.

**See Also**

*Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Process Scheduler*

*Enterprise PeopleTools 8.45 PeopleBook: Using PeopleSoft Applications*

# PeopleSoft Component Interfaces Preface

This book describes PeopleSoft Component Interfaces. It is written for programmers who will be accessing PeopleSoft components, usually using external systems.

---

## PeopleSoft Component Interfaces

A component interface is a PeopleSoft PeopleTools definition that you create in PeopleSoft Application Designer. It enables synchronous access to a PeopleSoft component from another application.



# CHAPTER 1

## Getting Started with PeopleSoft Component Interfaces

This chapter provides an overview of PeopleSoft Component Interfaces and discusses how to implement them.

---

### PeopleSoft Component Interfaces Technology Overview

A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft database information programmatically. PeopleSoft Component Interfaces expose a PeopleSoft component (a set of pages grouped for a business purpose) for synchronous access from another application (PeopleCode, Java, C/C++, or Component Object Model [COM]). A PeopleCode program or an external program (Java, C/C++, or COM) can view, enter, manipulate, and access PeopleSoft component data, business logic, and functionality. Additionally, you can use the Component Interface Tester to check the validity of your component interface and the Excel to Component Interface Utility to manage your data.

Component interfaces are created in PeopleSoft Application Designer, so you should ensure that you are familiar with PeopleTools and PeopleSoft Application Designer.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Application Designer*.

This section provides information to consider before you begin to use PeopleSoft Component Interfaces. In addition to implementation considerations presented in this section, take advantage of all PeopleSoft sources of information, including the installation guides, release notes, and PeopleBooks.

---

### Implementing PeopleSoft Component Interfaces

The functionality to create component interfaces for your applications is delivered as part of standard PeopleSoft PeopleTools that are provided with all PeopleSoft products.

You must complete these activities before you begin to create component interfaces for your implementation:

- Install your PeopleSoft application according to the installation guide for your database type.

See the PeopleSoft installation guide for your platform and product line.

- Establish a user profile that gives you access to PeopleSoft Application Designer and any other processes that you will use.

See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Understanding PeopleSoft Security”.

## **Implementing the Excel to Component Interfaces Utility**

There are several tasks involved in setting up the Excel to Component Interfaces Utility.

See Chapter 9, “Using the Excel to Component Interface Utility,” Getting Started with the Excel to Component Interface Utility, page 100.

## CHAPTER 2

# Introducing Component Interfaces

This chapter discusses:

- Component interface architecture.
- Component interface concepts, such as names and properties.
- Component interface definition in PeopleSoft Application Designer.

---

## Understanding Component Interfaces

A component interface enables exposure of a PeopleSoft component (a set of pages grouped together for a business purpose) for synchronous access from another application (such as PeopleCode, Java, C/C++, COM, or XML). Component interfaces can be viewed as "black boxes" that encapsulate PeopleSoft data and business processes, and hide the details of the underlying page and data. Component interfaces can be used to integrate one PeopleSoft application with another PeopleSoft application or with external systems. Component interfaces execute the business logic built into the component and as a result, they provide a higher level of data validation than a simple SQL insert.

A component interface maps to one, and only one, PeopleSoft component. However, you can create multiple component interfaces on top of the same component. You create component interfaces in PeopleSoft Application Designer. Record fields on the PeopleSoft component are mapped to the keys and properties of the component interface. Methods are used to find, create, modify or delete data.

---

## Component Interface Architecture

The component interface architecture comprises three fundamental elements—components, component interfaces, and the component interface API.

Every component interface has the following main attributes:

- Name
- Keys (get keys, create keys, and find keys)
- Properties and collections (fields and records)
- Methods

---

**Note.** In most cases, component interfaces behave like their associated components meaning that PeopleCode events typically trigger in the same order as the component. However, several runtime exceptions relate to component interfaces and PeopleCode processing and search dialog box processing.

---

## See Also

*Enterprise PeopleTools 8.45 PeopleBook: PeopleCode Developer's Guide*, "PeopleCode and the Component Processor"

---

# Component Interface Attributes

This section discusses the name, keys, properties, collections, and methods of component interfaces.

## Name

Each component interface requires a unique name that is specified when the component interface is created. The calling programs use the name of the component interface to access properties and methods.

## Keys

Keys are special properties containing values that retrieve an instance (get keys) or a list of instances (find keys) of the component interface. When you create a new component interface, get and find keys are created based on the search record definition for the underlying component. However, you can add, remove, or change keys in PeopleSoft Application Designer. Create keys get created for components that have the Add action enabled.

## Properties

Properties provide access to both component data and component interface settings. Component interfaces include two types of properties: standard and user-defined.

- Standard properties are assigned automatically when a component interface is created.  
Standard properties can be set to true or false. These properties are not displayed in the PeopleSoft Application Designer. Examples of standard properties include InteractiveMode, GetHistoryItems, and EditHistoryItems.
- User-defined properties map to record fields on the PeopleSoft component and are displayed in the PeopleSoft Application Designer.

A property can correspond to a field or a scroll (collection). You have control over which user-defined properties are included on the component interface.

---

**Note.** Every PeopleSoft Application Designer definition—including the component interface—has a definition properties dialog box in which you make design-time settings for the definition. Those properties should not be confused with the runtime properties that are discussed here.

---

## Collections

A component interface collection is a special type of property that corresponds to a scroll. It contains fields and subordinate scrolls as defined in its underlying component. By default, each collection uses the name of the primary record for the underlying scroll.

## Methods

A method is a function that performs a specific task on a component interface at runtime. As with component interface properties, there are two main types of methods: standard and user-defined. For example, you can use methods to save or create a new purchase order. Runtime access to each method is determined by the security that you have for that specific method.

- Standard methods are those that are available for all component interfaces.

The Find, Get, Save, and Cancel methods are automatically generated by PeopleSoft Application Designer when a new component interface is created. The Create method gets created for components that have the Add action enabled. In the component interface designer, standard methods are highlighted in gray.

- User-defined methods are created in PeopleSoft Application Designer to provide added functionality to the component interface.

These methods are functions that are made accessible through the component interface. Each function maps to a user-defined method. In the component interface designer, user-defined methods are highlighted in blue.

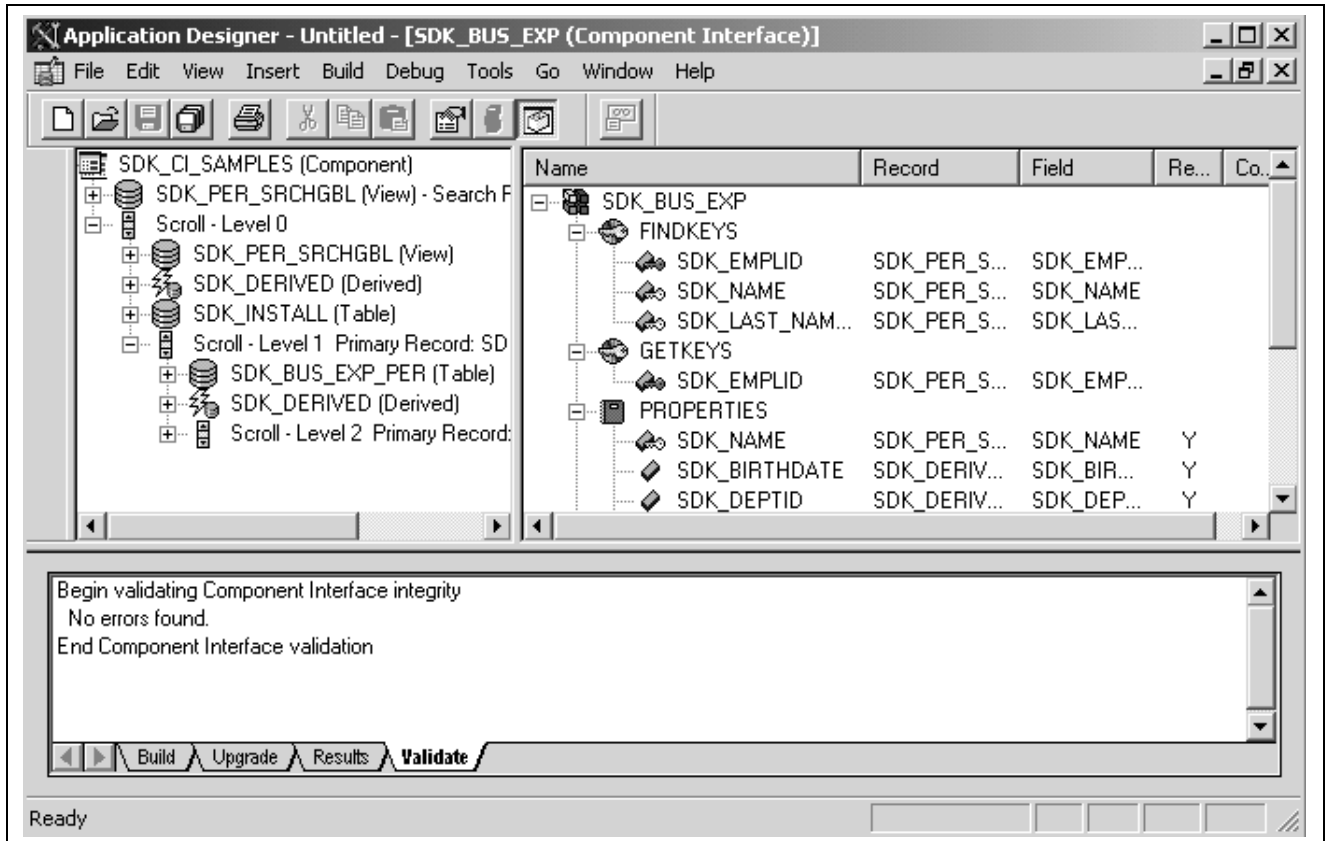
---

## Component Interface Definitions and Views

You create, modify, and review your component interface definition by using PeopleSoft Application Designer. You open the component interface definition just as you would any other definition, such as a page or record.

When working with a component interface definition in PeopleSoft Application Designer, you see the component view on the left and the component interface view on the right.

This screen shot shows the component and component interface view in PeopleSoft Application Designer.



Component and component interface views





The component view shows records and scrolls in the component, using a tree representation. The structure is the same as the one you see on the structure tab of a component in PeopleSoft Application Designer. Drag the fields and collections that you want exposed to the component interface view.

The component interface view shows the exposed keys, properties, and methods, using a tree representation. When you open a component interface, properties are displayed in the order in which they appear in the component view.

The tree nodes in both the component view and the component interface view have different icons. Some icons are used in both the component view and the component interface view with slightly different meanings. The following tables explain the meaning of each icon and column in the component interface view.

### Component Interface View Icons

Following is a list of the component interface view icons:

-  Component interface.
-  Group of keys.
-  Property that is a key field from the underlying record.
-  Alternate search key.



Group of properties or methods.



Collection.



Property or user-defined method.



Standard method.



Property that is a required field for the underlying record.



Item in a component interface that is no longer in sync with the underlying component. For example, if a field on which a property depends is deleted from the component, this icon appears.

## Component Interface View Columns

The following terms describe the columns in the component interface view.

<b>Name</b>	Name of a specific element of a component interface (such as the name of a property or method). The default name for field properties is the field name. The default name for collections is the primary record name.
<b>Record</b>	Name of the underlying record on which a specific element is based. If the underlying record name changes, the component interface continues to point to the appropriate record.
<b>Field</b>	Name of the field to which a component interface property points. Like the record name, the underlying field name can change, and the component interface continues to point to the appropriate field.
<b>Read Only</b>	Y in this column indicates that a specific property has been marked read-only.
<b>Comment</b>	Identifies comments that exist in the Edit Property dialog box for the selected key, property, or collection.

---

**Note.** In the component interface view, properties appear in the same order as they appear in the component and are not sorted alphabetically.

---



## CHAPTER 3

# Developing the Component Interface

This chapter discusses how to:

- Create a component interface definition.
- Use component interface keys, properties, collections, and methods.
- Validate the component interface.
- Set component interface security.
- Test the component interface.
- Understand runtime considerations.

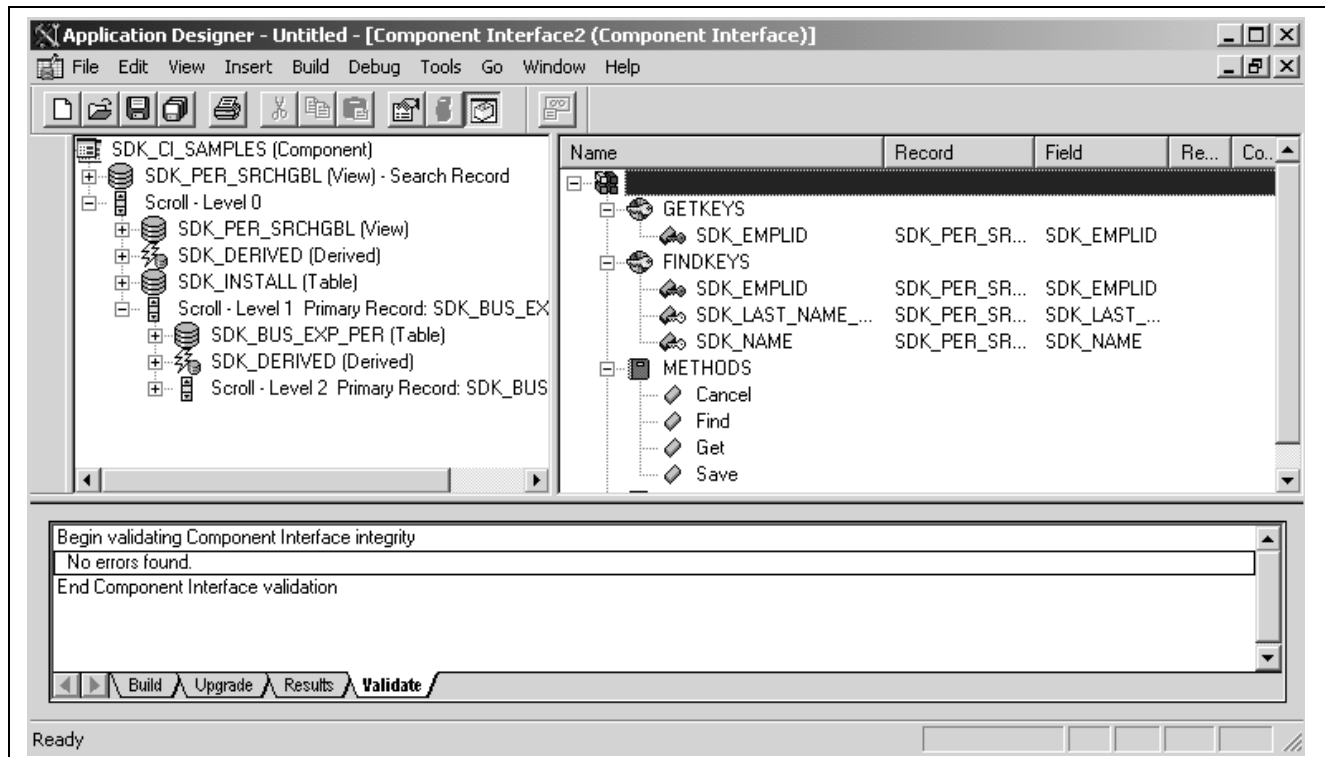
---

## Creating the Component Interface Definition

Because each component interface refers to a single component, you must know the structure of the component for which you're constructing a component interface. You can use an existing component within your application or create a new one for the sole purpose of constructing a component interface. Many parts of the component interface, such as the keys, are created based on settings in the referenced component.

### Creating a New Component Interface

This section discusses how to create a new component interface.



New component interface with no properties

To create a new component interface:

1. Select File, New from the PeopleSoft Application Designer menu.
2. Select Component Interface from the New dialog box.
3. Select the component on which this component interface will be based.

After you select the appropriate component, you see a message asking if you want the fields exposed in the selected component to become the default properties of the component interface.

---

**Note.** Not all fields on the component interface can have automatic defaults created for them.

---

4. Click *Yes* to confirm the default property definitions or *No* if you don't want any properties initially created.

If you elect to have the property definitions automatically defaulted by the system, then all properties that appear on the pages of the underlying component are added to the component interface. Even though the system adds the default properties, you may need to move other properties into the component view for the component to work.

An untitled component interface appears, showing the Get keys and Find keys. Create keys are produced only if the underlying component is able to run in Add mode (the example preceding this procedure does not have Create keys, because the search record of the underlying component cannot run in Add mode). PeopleSoft Application Designer generates the keys for you as you drag definitions.

The standard methods Cancel, Find, Get, and Save are automatically created. The Create method is not automatically created unless the component supports the Add mode.

---

**Note.** You can begin adding properties to a new component interface at any point. However, you cannot add any user-defined methods to the component interface until you have saved the component interface.

---

5. Save the component interface.

Once you have saved the component interface, you can further define user-defined methods.

## Criteria for Defaulting Properties

To be able to set automatic defaults for fields in the new component interface, the system needs the properties to be of a specific field or page control type.

The fields should be of the following types:

- Character
- Long character
- Number
- Signed number
- Date
- Time
- Datetime

The field should be one of the following page control types:

- Edit box
- Drop-down list box
- Check box
- Radio button

The field cannot be invisible and should not be the same as the immediate parent's key field.

Collections must have at least one child property that satisfies the field or page control criteria for defaulting the field. Collections with no properties are not added.

For a field on a secondary page to be selected for the default properties process, it must satisfy all the criteria for field type and page control and must be at the same level as the host page.

Additionally, the component tree that a component interface uses to order the properties lists the fields in the record based on their order in the record definition and not the order of the fields on the page. If the component tree lists the fields of a record based on the page, the properties of the component interface will reflect that order.

## Naming the Component Interface Definition

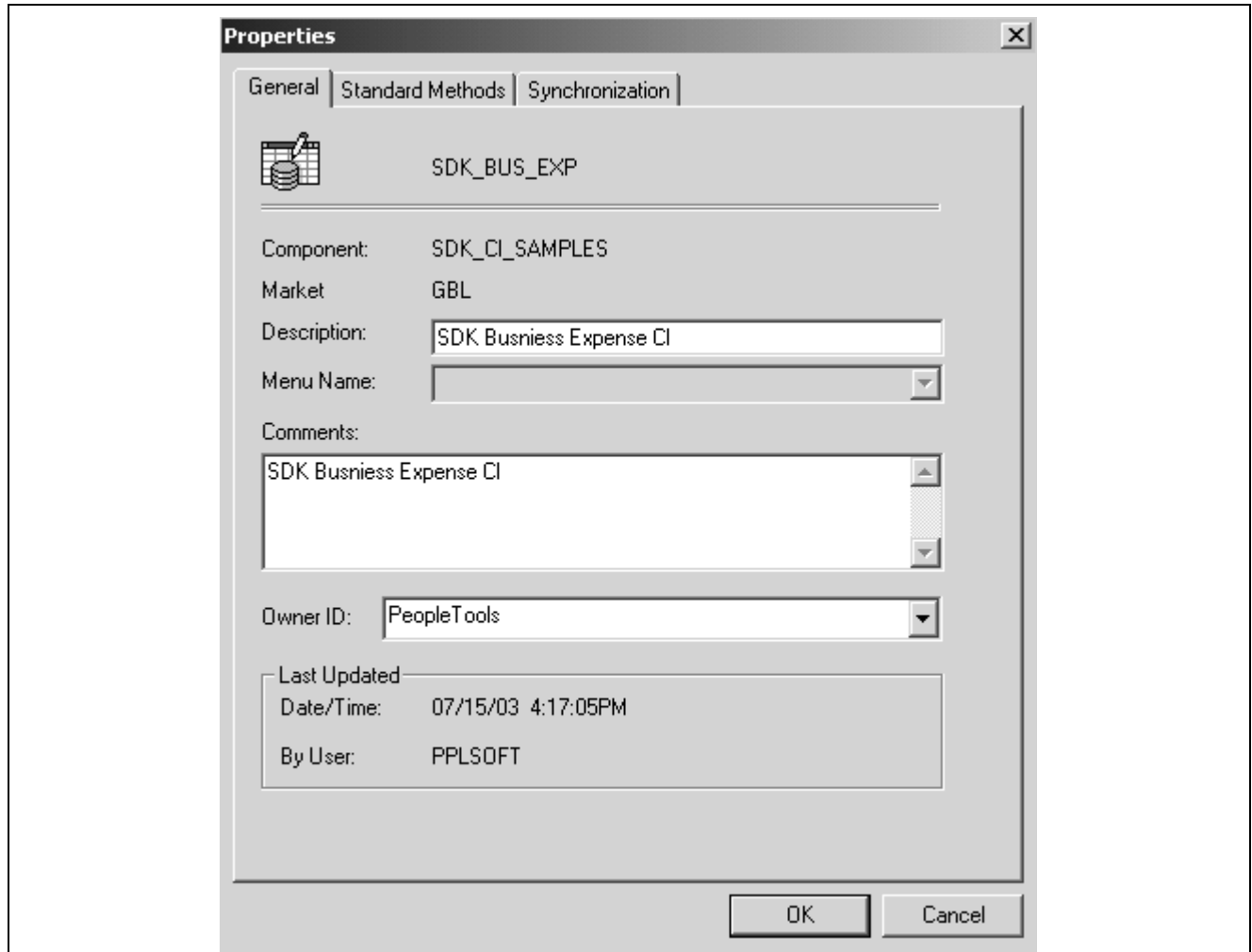
Like every other definition in PeopleTools, component interfaces must have unique names. The naming of component interfaces should be consistent and systematic. Also, the name should not be changed once the component interface is part of a production system—other applications depend on a consistent name with which to reference the component interface.

If you are changing the structure of a component interface such that an existing program can no longer access it correctly, create a new component interface rather than updating the existing one. There is no version property on a component interface, so if you must create a new version of a delivered component interface, adhere to a standard naming guideline to avoid confusion. A suggested naming guideline is:

- LOCATION (original component interface).
- LOCATION\_V2 (version two of the component interface).

## Associating a Component Interface with a Menu

This applies to component interfaces built from components that are already attached to one or more menus.



Component Interface Properties: General tab

To associate a component interface with a menu:

1. Select File, Open from the PeopleSoft Application Designer menu to open an existing component interface.
2. Select File, Definition Properties from the PeopleSoft Application Designer menu.  
The Definition Properties dialog box appears.
3. Select the appropriate menu name on the General tab for this component interface.

---

**Note.** Associate a menu with a component interface only when there is PeopleCode in the component that uses the %Menu system variable.

---

## Determining Which Fields to Expose

You expose fields from a component in the component interface by dragging a record field or a scroll from the component view into the component interface view. However, some forethought is required before exposing a component as a component interface. You need to have a thorough understanding of the underlying component so that you expose fields that are required in the external system. For example, if the component has a field called SSN, you need to be sure that the SSN field is required before exposing it to the external system. Expose only those properties that are necessary.

The component view displays fields that are available in the component buffer at runtime. For example, if a record containing 10 fields has only 3 fields included on a page, then the component view will list only those 3 fields.

The first time that you drag a scroll from the component view to the component interface view, the system uses the following rules to determine what properties to expose:

- Keys are exposed only at the highest-level collection in which they first appear.

In some cases, this is not appropriate. When an effective-dated component that has the same level-zero and level-one record is exposed through a component interface, it should be exposed the same way in which it appears on a page in the component. In this case, only one key field typically appears at level zero and the effective-date keys appear at level one. The component interface wrapper should expose the page in the same fashion—removing keys that do not appear in the level-zero scroll from the component interface top-level collection and manually adding keys that appear in level-one scroll to the second-level collection.

Typically, you do not want to expose Get keys or Create keys as properties, because these are set before a Get or Create operation and might be inadvertently changed.

- Make sure that you do not delete all the properties within the collection; that would result in an empty collection. If such empty collections exist, remove them; otherwise, they appear with X in the component interface view.
- If your page does not support Add mode, then you should not expose the level-zero record of the component, because it contains data that is not specific to the component interface that you are creating.
- Do not expose fields that are not visible in the component view.

The component optimization code might eliminate unused fields from its buffers, which results in an error when that field is accessed by the component interface.

---

## Using Keys

This section discusses how to add and delete keys.

### Understanding Keys

The following table shows the three types of component interface keys:

Key Type	Key Characteristics
Get keys	These keys automatically map to search key fields in the search record for the underlying component. You must change Get keys only if you modify the keys of the component after you create a component interface.
Find keys	These map to both search key fields and alternate search key fields in the search record for the underlying component. You may remove any Find keys based on alternate search key fields that you don't want to make available for searching.
Create keys	If the underlying component allows the Add action, then Create keys are generated for the component interface automatically. They map to fields marked as <i>Srch</i> (search) in the search record for the component (or the add search record, if one is specified).

Keys are created automatically when you create a component interface. Typically, you must manually add keys only if new search key fields are added to the underlying component after the creation of the component interface. However, you might want to modify the Find keys—either to restrict a user from searching on a particular key or to add an alternate search key that didn't exist when the component was created.

Component interface keys are based only on the search key fields and alternate search key fields that are designated as list box items in the search record of the underlying component. When you create the component interface, the keys are automatically generated from all key fields that qualify.

- Each search key field produces a Get key and a Find key.
- Each search key field also produces a Create key if the underlying component allows Add mode.
- Each alternate search key field produces a Find key.

### Valid Conditions for Modifying Keys

The following conditions are valid for modifying keys.

- You can add or delete a Find key if it's based on an alternate search key field.
- You can add any type of key based on a qualifying search key field in the component, if it isn't already the basis of an existing key of the same type.

This is necessary only if a new search key field is added to the component after you create the component interface.

- You can delete any type of key if its underlying search key field meets one of these criteria:
  - It is no longer defined as a search key field.
  - It is no longer designated as a list box item.
  - It has been deleted from the component.

---

**Note.** An *X* icon precedes a name in the component interface view if the field underlying a component interface key no longer qualifies as a key. Remove keys (or any other properties) that are marked with this symbol to ensure proper operation of the component interface.

---

## Adding and Deleting Keys

To add a key:

1. Expand the search key collection (the first collection) in the component view.
2. Drag the key to the component interface view.

To delete a key:

1. Select the key in the component interface view.
2. Press the Del key.

---

## Setting Properties

This section discusses how to:

- Create user-defined properties.
- Delete properties.
- Rename properties.
- Make properties read-only.
- Name component interface properties.

## Standard Properties

Standard properties do not appear in the component interface view in PeopleSoft Application Designer. These tables name and define the standard properties, and list the interfaces for PeopleCode, Java, C++, and Visual Basic.

This table contains the component interface properties:

Name	Description, Programming Syntax
CreateKeyInfoCollection	<p>Returns a collection of items that describes the Create keys. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: IcompIntfcPropertyInfoCollection getCreateKeyInfoCollection()</li> <li>• C++: HPSAPI_ COMPINTFCPROPERTYINFOCOLLECTION &lt;CI_NAME&gt;_GetCreateKeyInfoCollection (HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: CompIntfcPropertyInfoCollection CreateKeyInfoCollection</li> </ul>
GetKeyInfoCollection	<p>Returns a collection of items that describes the Get keys. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: IcompIntfcPropertyInfoCollection getGetKeyInfoCollection()</li> <li>• C++: HPSAPI_ COMPINTFCPROPERTYINFOCOLLECTION &lt;CI_NAME&gt;_GetGetKeyInfoCollection(HPSAPI_ &lt;CI_NAME&gt;)</li> <li>• COM: CompIntfcPropertyInfoCollection GetKeyInfoCollection</li> </ul>
FindKeyInfoCollection	<p>Returns a collection of items that describes the Find keys. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: IcompIntfcPropertyInfoCollection getFindKeyInfoCollection()</li> <li>• C++: HPSAPI_ COMPINTFCPROPERTYINFOCOLLECTION &lt;CI_NAME&gt;_GetFindKeyInfoCollection(HPSAPI_ &lt;CI_NAME&gt;)</li> <li>• COM: CompIntfcPropertyInfoCollection FindKeyInfoCollection</li> </ul>

Name	Description, Programming Syntax
GetHistoryItems	<p>Controls whether the component interface runs in Update/Display mode or Correction mode when the underlying component is effective dated. If GetHistory is set to true, then historical data can be retrieved but not modified. GetHistory items work in accordance with EditHistory items.</p> <p>The default value is False. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean getGetHistoryItems(), void setGetHistoryItems(boolean)</li> <li>• C++: BOOL &lt;CI_NAME&gt;_GetGetHistoryItems(HPSAPI_&lt;CI_NAME&gt;), void &lt;CI_NAME&gt;_SetGetHistoryItems(HPSAPI_&lt;CI_NAME&gt;, BOOL)</li> <li>• COM: Boolean GetHistoryItems</li> </ul>
EditHistoryItems	<p>Controls whether the component interface runs in Update/Display All mode, Update/Display mode, or Correction mode when the underlying component is effective dated. If EditHistory items are set to true, then historical data can be modified. EditHistory items work in accordance with GetHistory items.</p> <p>The default value is False. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean getEditHistoryItems(), void setEditHistoryItems(boolean)</li> <li>• C++: BOOL &lt;CI_NAME&gt;_GetEditHistoryItems(HPSAPI_&lt;CI_NAME&gt;), void &lt;CI_NAME&gt;_SetEditHistoryItems(HPSAPI_&lt;CI_NAME&gt;, BOOL)</li> <li>• COM: Boolean EditHistoryItems</li> </ul>

Name	Description, Programming Syntax
<p>InteractiveMode</p>	<p>Controls whether to apply values and run business rules immediately, or whether items are queued and business rules are run later, in a single step.</p> <p><b>Note.</b> You should use interactive mode when testing and debugging a component interface. Interactive mode in a production environment slows performance because of the number of server trips required.</p> <p>If you're using a component interface as part of a batch process in which thousands of rows are to be inserted, running in interactive mode may reduce performance so much on some UNIX servers that the application times out with a connection failure.</p> <p>The default value is False. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean getInteractiveMode(), void setInteractiveMode(boolean)</li> <li>• C++: BOOL &lt;CI_NAME&gt;_GetInteractiveMode (HPSAPI_&lt;CI_NAME&gt;), void &lt;CI_NAME&gt;_SetInteractiveMode(HPSAPI_&lt;CI_NAME&gt;, BOOL )</li> <li>• COM: Boolean InteractiveMode</li> </ul>
<p>StopOnFirstError</p>	<p>When this property is set to True, the first error generated by the component interface halts the program.</p> <p>The default value is False. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getStopOnFirstError(), setStopOnFirstError(boolean)</li> <li>• C++: BOOL &lt;CI_NAME&gt;_GetStopOnFirstError (HPSAPI_&lt;CI_NAME&gt;), void &lt;CI_NAME&gt;_SetStopOnFirstError(HPSAPI_&lt;CI_NAME&gt;, BOOL)</li> <li>• COM: Boolean StopOnFirstError</li> </ul>
<p>CompIntfcName</p>	<p>Returns the name of the component interface class as named in PeopleSoft Application Designer. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getCompIntfcName()</li> <li>• C++: LPTSTR &lt;CI_NAME&gt;_GetCompIntfcName( HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: String GetCompIntfcName</li> </ul>

Name	Description, Programming Syntax
ComponentName	<p>Returns the name of the component interface class as named in Application Designer. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getComponentName()</li> <li>• C++: LPTSTR &lt;CI_NAME&gt;_GetComponentName(HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: Boolean GetComponentName</li> </ul>
Description	<p>Returns the description of the component interface class as set in Application Designer. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getDescription()</li> <li>• C++: LPTSTR &lt;CI_NAME&gt;_GetDescription(HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: String Description</li> </ul>
Market	<p>Returns the Market setting of the component used to build this component interface. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getMarket()</li> <li>• C++: LPTSTR &lt;CI_NAME&gt;_GetMarket(HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: String Market</li> </ul>

Name	Description, Programming Syntax
GetDummyRows	<p>When a new scroll is inserted on a page, that scroll is displayed even though it has no underlying data. Any scroll that is empty has one dummy row displayed with only the defaults set. This property is True if the dummy row is to be displayed, False if you do not want to display the dummy row. The default value for this property is True. This property is read-write.</p> <ul style="list-style-type: none"> <li>• Java: boolean getGetDummyRows(), void setGetDummyRows(boolean)</li> <li>• C++: BOOL &lt;CI_NAME&gt;_GetGetDummyRows (HPSAPI_&lt;CI_NAME&gt;), void &lt;CI_NAME&gt;_SetGetDummyRows(HPSAPI_&lt;CI_NAME&gt;, BOOL)</li> <li>• COM: Boolean GetDummyRows</li> </ul>
PropertyInfoCollection	<p>Returns a collection of items that describes a specific property. The specific properties that are available in the propertyinfocollection are listed here. This property is read-only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: IcompIntfcPropertyInfoCollection getPropertyInfoCollection()</li> <li>• C++: HPSAPI_ COMPINTFCPROPERTYINFOCOLLECTION &lt;CI_NAME&gt;_GetPropertyInfoCollection(HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: CompIntfcPropertyInfoCollection PropertyInfoCollection</li> </ul>

The CompIntfPropInfoCollection object supports the following properties:

PropertyName	Description
Name	<p>This property returns the name of the object executing the property as a string. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getName()</li> <li>• C++: LPTSTR CompIntfcPropertyInfo_GetName (HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: String name</li> </ul>

PropertyName	Description
RecordName	<p>This property returns the Record Name associated with the object executing the property. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getRecordName()</li> <li>• C++: LPTSTR CompIntfcPropertyInfo_GetRecordName(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: String RecordName</li> </ul>
FieldName	<p>This property returns the Field Name associated with the object executing the property. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getFieldName()</li> <li>• C++: LPTSTR CompIntfcPropertyInfo_GetFieldName(HPSAPI_COMPINTFCPROPERTYINFO \)</li> <li>• COM: String FieldName</li> </ul>
LabelLong	<p>This property returns the record field LongName value as a string. If there is a component override for this value, it is not included. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getLabelLong()</li> <li>• C++: LPTSTR CompIntfcPropertyInfo_GetLabelLong(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: String LabelLong</li> </ul>
LabelShort	<p>This property returns the record field ShortName value as a string. If there is a component override for this value, it is not included. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getLabelShort()</li> <li>• C++: LPTSTR CompIntfcPropertyInfo_GetLabelShort(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: String LabelShort</li> </ul>
IsCollection	<p>This property returns True if the object executing the property is a data collection, False otherwise. If IsCollection is True, other field-oriented properties like Required, Type, Xlat, YesNo, Prompt and Format are undefined. If IsCollection is False, the object represents a field and all the previous properties are defined as described. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getIsCollection()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetIsCollection(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean IsCollection</li> </ul>

PropertyName	Description
Type	<p>This property returns the field type, as a number, of the object. Complete list of values can be found in (Link to the PeopleCode book &gt; PeopleCode Reference &gt; Component Interface Classes &gt; CompIntfPropInfoCollection Object Properties ). This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: long getType()</li> <li>• C++: PSI32 CompIntfcPropertyInfo_GetType(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Long Type</li> </ul>
OAType	<p>This property returns the field type, as a number, of the object. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: long getOAType()</li> <li>• C++: PSI32 CompIntfcPropertyInfo_GetOAType(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Long OAType</li> </ul>
Format	<p>This property returns the field format for the object executing the property (that is, name, phone, zip, SSN, and so on) as a number. This property is read-only.</p> <p><i>See Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference, "Component Interface Classes," CompIntfPropInfoCollection Object Properties.</i></p> <ul style="list-style-type: none"> <li>• Java: String getFormat()</li> <li>• C++: PSI32 CompIntfcPropertyInfo_GetFormat(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Long Format</li> </ul>
Key	<p>This property returns True if the object executing the property is a key, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getKey()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetKey(HPSAPI_COMPINTFCPROPERTYINFO hCompIntfcPropertyInfo)</li> <li>• COM: Boolean Key</li> </ul>
Required	<p>This property returns True if the object executing the property is a required property, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getRequired()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetRequired(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean Required</li> </ul>

PropertyName	Description
Xlat	<p>This property returns True if the object executing the property is associated with an XLAT table, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: String getXlat()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetXlat(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: String Xlat</li> </ul>
Yesno	<p>This property returns True if the object executing the property is associated with the Yes/No table, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getYesno()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetYesno(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean Yesno</li> </ul>
Prompt	<p>This property returns True if the object executing the property is associated with a prompt table, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getPrompt()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetPrompt(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean Prompt</li> </ul>
Length	<p>This property returns the length of the object executing the property. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: long getLength()</li> <li>• C++: PSI32 CompIntfcPropertyInfo_GetLength(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Long Length</li> </ul>
DecimalPosition	<p>This property returns the decimal position for the object executing the property. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: long getDecimalPosition()</li> <li>• C++: PSI32 CompIntfcPropertyInfo_GetDecimalPosition(HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Long DecimalPosition</li> </ul>

PropertyName	Description
IsReadOnly	<p>This property returns True if the property marked read-only in the component interface definition; False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getIsReadOnly()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetIsReadOnly (HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean IsReadOnly</li> </ul>
Altkey	<p>This property returns True if the object executing the property is an alternate key, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getAltkey()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetAltkey (HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean Altkey</li> </ul>
Listboxitem	<p>This property returns True if the object executing the property is associated with a list box, False otherwise. This property is read-only.</p> <ul style="list-style-type: none"> <li>• Java: boolean getListboxitem()</li> <li>• C++: BOOL CompIntfcPropertyInfo_GetListboxitem (HPSAPI_COMPINTFCPROPERTYINFO)</li> <li>• COM: Boolean Listboxitem</li> </ul>

### Example for PropertyInfoCollection

Here is a Java example that calls PropertyInfoCollection:

```

IcompIntfcPropertyInfoCollection oLO_PropInfoColl
IcompIntfcPropertyInfo oLO_PropInfoItem

oLO_PropInfoColl = oCI.getPropertyInfoCollection();
for (int I=0; I < oLO_PropInfoColl.getCount(); I++) {
    oLO_PropInfoItem = oLO_PropInfoColl.item(i);

    System.out.println("\t Name = " + oLO_PropInfoColl.getName());
    System.out.println("\t Record Name = " + oLO_PropInfoColl.getRecordName());
    System.out.println("\t Field Name = " + oLO_PropInfoColl.getFieldName());
    System.out.println("\t Label Long = " + oLO_PropInfoColl.getLabelLong());
    System.out.println("\t Label Short = " + oLO_PropInfoColl.getLabelShort());
    System.out.println("\t IsCollection = " + oLO_PropInfoColl.getIsCollection());
    System.out.println("\t Type = " + oLO_PropInfoColl.getType());
    System.out.println("\t OAType = " + oLO_PropInfoColl.getOAType());
    System.out.println("\t Format = " + oLO_PropInfoColl.getFormat());
    System.out.println("\t Is Get Key? = " + oLO_PropInfoColl.getKey());
    System.out.println("\t Is Required = " + oLO_PropInfoColl.getRequired());
    System.out.println("\t Is Xlat? = " + oLO_PropInfoColl.getXlat());

```

```

System.out.println("\t Is Yesno? = " + oLO_PropInfoColl.getYesno());
System.out.println("\t Prompt = " + oLO_PropInfoColl.getPrompt());
System.out.println("\t Length = " + oLO_PropInfoColl.getLength());
System.out.println("\t DecimalPosition = " + oLO_PropInfoColl.getDecimal⇒
Position());
System.out.println("\t Is Read Only? = " + oLO_PropInfoColl.getIsReadOnly());
System.out.println("\t Is Alt Key? = " + oLO_PropInfoColl.getAltkey());
System.out.println("\t Is ListBox item? = " + oLO_PropInfoColl.getListboxitem⇒
());

```

## Object Adapter

The name of the property is `OAType`, and it holds the value of the object adapter type. Exposing this property and supplying the associated methods enables you to detect possible data type mismatches between the database and the component interface object.

The Java methods are:

**getOAType()** Returns the object adapter type.

**getType()** Returns the type of the property of a particular database field.

For example:

```

public static void printPropertyType(String propName, ICompIntfcPropertyInfo i⇒
PropertyInfo) {

String strOAType = null;
String strDBType = null;

try {
switch ((int)iPropertyInfo.getOAType()) {
/* Object Adapter Type == 0 */
case CIPROPERTYTYPES.PSPROPERTY_OA_TYPE_BOOL:
strOAType = "BOOL";
break;
/* Object Adapter Type == 1 */
case CIPROPERTYTYPES.PSPROPERTY_OA_TYPE_NUMBER:
strOAType = "INTEGER";
break;
/* Object Adapter Type == 2 */
case CIPROPERTYTYPES.PSPROPERTY_OA_TYPE_FLOAT:
strOAType = "FLOAT";
break;
/* Object Adapter Type == 3 */
case CIPROPERTYTYPES.PSPROPERTY_OA_TYPE_STRING:
strOAType = "STRING";
break;
}

switch ((int)iPropertyInfo.getType()) {

```

```

        /* Database Type == 0 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_CHARACTER:
            strDBType = "CHARACTER";
            break;
        /* Database Type == 1 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_LONG_CHARACTER:
            strDBType = "LONG_CHARACTER";
            break;
        /* Database Type == 2 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_NUMBER:
            strDBType = "NUMBER";
            break;
        /* Database Type == 3 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_SIGNED_NUMBER:
            strDBType = "SIGNED_NUMBER";
            break;
        /* Database Type == 4 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_DATE:
            strDBType = "DATE";
            break;
        /* Database Type == 5 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_TIME:
            strDBType = "TIME";
            break;
        /* Database Type == 6 */
        case CIPROPERTYTYPES.PSPROPERTY_DB_TYPE_DATETIME:
            strDBType = "DATETIME";
            break;
    }

}

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

System.out.println("\n" + propName +
    " Object Adapter Type is: " + strOAType +
    ", Database Type is: " + strDBType);
}

```

## Component Interface Collection Property

This table contains the component interface collection property Count.

Name	Description, Programming Syntax
Count	Returns the number of items in a collection. <ul style="list-style-type: none"> <li>• Java: long getCount()</li> <li>• C++: PSI32 CompIntfcCollection_GetCount (HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: Integer Count</li> </ul>

### Data Item Property

This table contains the data item property ItemNum:

Name	Description, Programming Syntax
ItemNum	Returns the position of the row within the given collection of a DataRow. <ul style="list-style-type: none"> <li>• Java: long getItemNum()</li> <li>• C++: PSI32 &lt;CI_NAME&gt;_GetItemNum(HPSAPI_&lt;CI_NAME&gt;)</li> <li>• COM: Integer ItemNum</li> </ul>

---

**Note.** The Component Interface classes contain information about PropertyInfo properties and related PeopleCode.

---

## Creating User-Defined Properties

User-defined properties are those properties on the underlying component that are exposed through the component interface. User-defined properties are derived from component with which the component interface is associated and must be added manually. They are the specific record fields that you expose to an external system with the component interface. You create user-defined properties in addition to the standard properties to enable data manipulation of the component. When you create a new component interface, if you accept the default properties, user-defined properties are created automatically for each field displayed to the user on the underlying component.

User-defined properties are the points where the component and the underlying database are exposed to the external system. This is the means that component interfaces use to add or change fields and data in the database.

To create a user-defined property, drag a record, field, or scroll from the component view to the component interface view.

It does not matter where you insert the definition in the component interface view. When the component interface is opened, the system automatically converts the field or record into a component interface property and places it in the appropriate place in the list of properties. Also, when you drag a definition from the component view into the component interface view, all “child” definitions are brought into the component interface automatically. Once these child properties are added to the component interface, you can remove each property individually, if desired.

Dragging a key from the search records, which precede the level-zero record in the page view, adds a key to all appropriate key collections (Get, Create, and Find) in the component interface. Because appropriate keys are added automatically when a component interface is first created, you typically must add keys only if the new keys are added to the underlying component after the creation of the component interface.

## Deleting User-Defined Properties

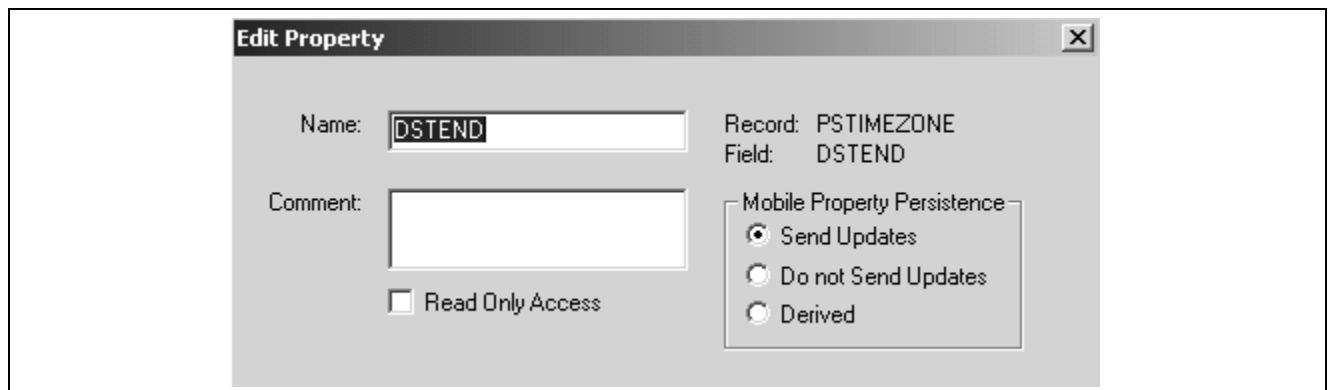
To delete a property:

1. Select the property to be deleted.
2. Either press the Del key on the keyboard, or right-click the key and select Delete.

Standard Windows behavior is employed for selecting multiple properties using the SHIFT and CTRL keys.

## Renaming User-Defined Properties

Property names are automatically generated according to the corresponding fields from the component. If these names are cryptic, you might want to rename these properties to explain them better. Renaming a property does not change the field that the property references.



Edit Property dialog box

To rename a property:

1. Double-click the property name or right-click the property name and select Edit Name from the pop-up menu.
2. Enter the new property name.

Programs accessing this component interface must reference the new property name. For example, if SDK\_NAME was changed to NAME, programs must use NAME instead of SDK\_NAME.

3. Add any comments that might be helpful.
4. Select the Read-Only check box to make this property read-only.
5. If this property is for a mobile application, select a radio button that sets the persistence of the property.

- Send Updates is the default behavior for a mobile property.  
Any changes or additions to this property on a mobile instance are synchronized to the server.
- If a mobile property is set to Do not Send Updates, this property is not synchronized up to the server, but the value is maintained on the device.
- A Derived property is used only at mobile runtime. Any values set or added to this property exist only for the runtime life of the object. There is no persistence of this data on the device, so it is subsequently never uploaded to the server.

---

**Note.** Application Designer generates an error message if it detects that a component interface has properties that resolve to the same name when creating, saving, or opening a given component interface.

For example, NAME1 and NAME\_1 both resolve to the same name when PeopleSoft APIs are built. The set and get functions that are generated for the properties RTE\_CNTL\_TYPE1 and RTE\_CNTL\_TYPE\_1 are:

```
public String getRteCntlType1()

public void setRteCntlType1(String inRteCntlType1)
```

This results in a compile error. To fix this condition, name the properties so that they do not resolve to the same name.

---

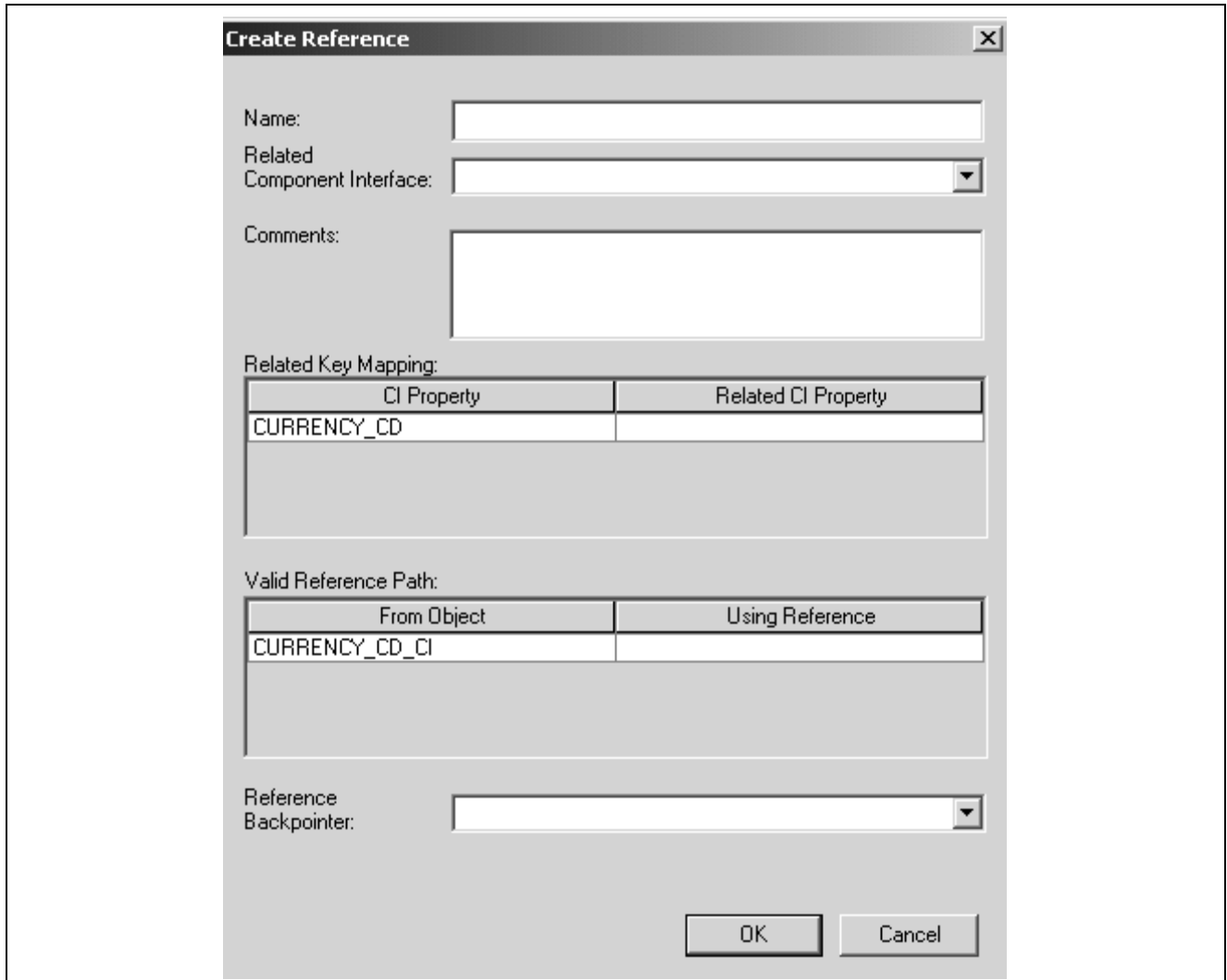
## Creating Reference Properties

Each component interface is isolated and unaware of the other component interfaces in the system. In order to access and update information from other component interfaces, references establish relationships between component interfaces.

Create a reference property in one component interface to access data exposed in another component interface. For example, the Customer object and the component interface exposing its properties include properties such as the customer's name, address, and telephone numbers. Another object, Contact, includes data associated with all contacts in the system. The link between a specific customer and its associated contacts is owned by the Contact record, not the Customer record.

Therefore, to access contact data, the Customer component interface needs a reference property referring to the Contact component interface. To update contact data from the Customer component interface, the reference must include a valid reference path and reference backpointer to the customer ID.

Access the Create Reference dialog box by right-clicking the property and selecting Create Reference.



Create Reference dialog box

The Create Reference dialog box has the following fields:

- Name** Describes the name of the reference you are creating.
- Related Component Interface** Designates the component interface referenced from the current component interface.
- Comments** Enter any comments to track the reference.
- Related Key Mapping** Maps the property from the related component interface to the selected component interface property.
- Valid Reference Path** Supports dynamic enumeration of the objects that can be selected as the value of the reference property being defined. This effectively filters these values so that you can select only objects that support the defined reference.  
  
Because references use the concept of a *walkpath* to go from level zero of one component interface to level zero of another component interface, and then “walk” down to the lower levels of the component interface, only the level zero references are displayed in the Valid Reference Path drop-down list of a reference definition.

**Reference Backpointer** Refers to the path back to the original component interface.

## Making Properties Read-Only

You can make any property read-only. At runtime, the value of a read-only property can be read but not updated.

To make a property read-only:

1. Select the property.
2. Select Edit, Toggle Read Only Access from the PeopleSoft Application Designer menu.

A *Y* appears in the Read Only column of the component interface view corresponding to each property that you selected to be read-only.

---

**Note.** You can double-click the icon of any existing user-defined property to edit its name or comment or to toggle read-only access.

---

---

## Working with Collections

A collection is a property that points to a scroll, rather than a field, in the underlying component for a component interface. A collection groups multiple fields in a scroll. All the fields in the scroll are mapped to a property. These properties are part of the collection.

You create collections the same way you create properties—drag the scroll from the component view into the component interface view. Consider these points when creating collections:

- When dragging a scroll into the component interface view, all child scrolls come with it.

This is the same behavior that you would expect when creating a property. Child properties are always added automatically when you drag a field from the component view to the component interface view. After the property or collection has been created, you can delete individual child properties or collections manually, if necessary.

- When dragging a scroll into the component interface view, all record fields contained in that scroll come with it—not just those from the record that defines the scroll.

The fields from all records at that scroll level are exposed as part of the same collection.

- Keys that appear in parent and child scrolls are not added to child collections.

For the component interface to function as expected, the keys must remain synchronized at all levels of the component. Having keys at lower levels makes it possible to compromise this synchronization. Therefore, lower-level keys are not introduced into the component interface and are not exposed to the user because those keys have already been set at the parent level.

- When you drag a child scroll into the component interface view, parent collections are created automatically.

For example, if you drag just the level-two scroll from the component view into the component interface view, a level-zero collection and a level-one collection are created automatically in the component interface. This hierarchy of collections is necessary so that it's possible to navigate to the child collection at runtime.

---

## Working with Methods

This section discusses how to:

- Enable and disable standard methods.
- Create user-defined methods.

## Working with Session Functions and Methods

The session functions and methods connect to a session on a PeopleSoft application server. This connection must be made before you can use the component interface methods.

### Component Interface Session Functions

This table contains the component interface session function createSession:

Name	Description, Programming Syntax
createSession (In PeopleCode, &session = %session)	Returns a session object. <ul style="list-style-type: none"> <li>• Java: ISession API.createSession()</li> <li>• C++: HPSAPI_SESSION PSApiCreateSession()</li> <li>• COM: PeopleSoft_PeopleSoft.Session CreateObject("PeopleSoft.Session")</li> </ul>

### Component Interface Session Methods

This table contains the component interface session methods:

Name	Description, Programming Syntax
<p>Connect</p> <p>(not used in PeopleCode)</p>	<p>Connects to the application server.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean connect(long apiVersion, string server, string username, string password, byte[ ] ExternalAuth)</li> <li>• C++: Bool session_Connect(HPSAPI hSession, PSI32 ApiVersion, LPTSTR server, LPSTR username, LPTSTR password, PSAPIVAPBLOB ExternalAuth)</li> <li>• COM: connect(apiVersion As Long, server As string, username As string, password As String, externalAuth As Integer) As Boolean</li> </ul>
<p>getCompIntfc</p>	<p>Returns a reference to a Component Interface. getCompIntfc also checks to see if the given user that is connecting has the appropriate security to access the component interface.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: I&lt;CI_Name&gt; getCompIntfc(string ciName)</li> <li>• C++: HPSAPI_&lt;CI_Name&gt; Session_GetCompIntfc (HPSAPI_SESSION hsession, LPTSTR ciName)</li> <li>• COM: &lt;CI_Name&gt; GetCompIntfc(ciName As String)</li> </ul>

## Standard Methods

A method is a definition that performs a specific function on a component interface at runtime. Each standard method is added by default when the component interface is created and is available in PeopleCode and other programming languages. Like properties, methods are saved as part of a component interface definition. There are two main types of methods: standard methods and user-defined methods.

Standard Methods	Description, Programming Syntax
Cancel	<p>Backs out of the current component interface, canceling any changes made since the last save. Equivalent to clicking the Return to Search button online. Returns True on success, and False on failure.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean cancel()</li> <li>• C++: BOOL &lt;CI_NAME&gt;_Cancel(HPSAPI_&lt;CI_NAME&gt; hObj)</li> <li>• COM: Function Cancel() As Boolean</li> </ul>
Create	<p>Creates a new instance of a component interface. Equivalent to creating a new record in Add mode online. Returns True on success, and False on failure.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean create()</li> <li>• C++: BOOL &lt;CI_NAME&gt;_Create(HPSAPI_&lt;CI_NAME&gt; hObj)</li> <li>• COM: Function Create() As Boolean</li> </ul>
Find	<p>Performs a partial key search for a particular instance of a component interface, using the search keys at level 0. Returns a collection of component interface instances which match the search criteria. If no component interface instances match the search criteria, the count on the collection is zero.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CI_NAME&gt;Collection find()</li> <li>• C++: HPSAPI_&lt;CI_NAME&gt;COLLECTION &lt;CI_NAME&gt;_Find(HPSAPI_&lt;CI_NAME&gt; hObj)</li> <li>• COM: Function Find() As &lt;CI_NAME&gt;Collection</li> </ul>

Standard Methods	Description, Programming Syntax
Get	<p>Retrieves a particular instance of a component interface. Equivalent to opening a record in Update/Display or Correction mode when online with a PeopleSoft application. Returns True on success, and False on failure.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean get()</li> <li>• C++: BOOL &lt;CI_NAME&gt;_Save(HPSAPI_&lt;CI_NAME&gt; hObj)</li> <li>• COM: Function Get() As Boolean</li> </ul>
Save	<p>Saves an instance of a component interface. Equivalent to clicking the Save button in the online system. Returns True on success, and False on failure. You should do a save after a cancel.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean save()</li> <li>• C++: BOOL &lt;CI_NAME&gt;_Save(HPSAPI_&lt;CI_NAME&gt; hObj)</li> <li>• COM: Function Save() As Boolean</li> </ul>
GetPropertyByName	<p>Returns the value of a property that is specified by name. This function typically is used only in applications that cannot get the names of the component interface properties until runtime.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: Object getPropertyByName(String str)</li> <li>• C++: HPSAPI_OBJECT &lt;CiCollectionItem&gt;_GetPropertyByName(HPSAPI_&lt;CI_COLLECTION_ITEM&gt; hCollItem, LPTSTR Name)</li> <li>• COM: Function GetPropertyByName(name As String)</li> </ul>

Standard Methods	Description, Programming Syntax
<p>SetPropertyByName</p>	<p>Sets the value of a property that is specified by name. This function typically is used only in applications that cannot set the names of the component interface properties until runtime.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: long setPropertyByName(String str, Object o)</li> <li>• C++: PSI32 &lt;CiCollectionItem&gt;_SetPropertyByName (HPSAPI_&lt;CI_COLLECTION_ITEM&gt; hCollItem, LPTSTR name, HPSAPI_OBJECT Value)</li> <li>• COM: Function SetPropertyByName(name As String, value) As Long</li> </ul>
<p>GetPropertyInfoByName (In PeopleCode, CompIntfPropInfoCollection)</p>	<p>Returns specific information, such as length, about the definition of a property that is specified by name. This function typically is used only in applications that cannot get the names of component interface properties until runtime or by applications that need to provide a dynamic list of values that would normally be found in prompt tables.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: IcompIntfcPropertyInfo getPropertyInfoByName(String name)</li> <li>• C++: HPSAPI_ COMPINTFCPROPERTYINFO&lt;CiPropOrItem&gt;_ GetPropertyInfoByName(HPSAPI_ &lt;CIPROPORITEM&gt; hPropOrItem, LPTSTR name) where CiPropOrItem is the name of either a property or an item in a collection.</li> <li>• COM: Function GetPropertyInfoByName(name As String) As CompIntfcPropertyInfo</li> </ul> <p>See <i>Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference</i>, “Component Interface Classes,” CompIntfPropInfoCollection Object Properties.</p>

By default, each component interface is created with four standard methods—Cancel, Find, Get, and Save. Additionally, the Create standard method is generated if Create keys have been added to the component interface.

### Example for GetPropertyInfoByName

The GetPropertyInfoByName method returns an object containing the property information. Here is a Java example that calls GetPropertyInfoByName:

```
IcompIntfcPropertyInfo oCompIntfcPropertyInfo
oCompIntfcPropertyInfo = oCI.getPropertyInfoByName(tempName);
```

```

System.out.println(oCompIntfcPropertyInfo.getName());
if (!oCompIntfcPropertyInfo.getIsCollection()) {
    System.out.println("\t Format = " + oCompIntfcPropertyInfo.getFormat());
    System.out.println("\t Type = " + oCompIntfcPropertyInfo.getType());
}
System.out.println("\t Is Required = " + oCompIntfcPropertyInfo.getRequired());
System.out.println("\t Is Collection? = " + oCompIntfcPropertyInfo.getIsCollection⇒
());
System.out.println("\t Is Read Only? = " + oCompIntfcPropertyInfo.getIsReadOnly());
System.out.println("\t Is Get Key? = " + oCompIntfcPropertyInfo.getKey());
System.out.println("\t Label Long = " + oCompIntfcPropertyInfo.getLabelLong());
System.out.println("\t Label Short = " + oCompIntfcPropertyInfo.getLabelShort());
System.out.println("\t Length = " + oCompIntfcPropertyInfo.getLength());
System.out.println("\t Name = " + oCompIntfcPropertyInfo.getName());
System.out.println("\t Is Xlat? = " + oCompIntfcPropertyInfo.getXlat());
System.out.println("\t Is Yesno? = " + oCompIntfcPropertyInfo.getYesno());

```

---

**Note.** When creating a new component interface, you must save the component interface before the standard methods are created. PeopleSoft Application Designer adds the standard methods upon the first save of a new component interface.

---

## Collection Methods

The first item in a component interface collection is always indexed as item 1 from PeopleCode and COM programs, which is consistent with other PeopleCode processing. From Java and C++ programs, this item is indexed as item 0.

### Component Interface Collection Properties

This table contains the component interface collection properties:

Data Collection Method	Action, Usage
Count	<p>Returns the number of items in a collection.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: long getCount()</li> <li>• C++: PSI32 &lt;CiCollectionName&gt;_GetCount (HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol)</li> <li>• COM: Count As Long</li> </ul>

Data Collection Method	Action, Usage
<p>ItemByName (not used in PeopleCode)</p>	<p>Returns the property in the collection. It takes <i>Name</i> as a parameter.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: ICompIntfcPropertyInfo itemByName(String Name)</li> <li>• C++: CompIntfcPropertyInfoCollection_ ItemByName (HPSAPI_ COMPINTFCPROPERTYINFOCOLLECTION, LPTSTR Name)</li> <li>• COM: Function ItemByName(name As String) As CompIntfcPropertyInfo</li> </ul>
<p>InsertItem(Index)</p>	<p>Inserts a new item. This is equivalent to clicking the Add button to insert a new row when online. It takes <i>Index</i> as a parameter and follows the same conventions for performing business rules (PeopleCode) as the online system.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CiCollectionName&gt; insertItem( long Index)</li> <li>• C++: HPSAPI_&lt;CI_COLLECTION_ITEM&gt; &lt;CiCollectionName&gt;_InsertItem(HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, PSI32 Index)</li> <li>• COM: Function InsertItem(index As long) As &lt;CI_COLLECTION_ITEM&gt;</li> </ul>
<p>DeleteItem(Index)</p>	<p>Deletes the item that is designated by <i>Index</i>. This is equivalent to clicking the Delete button to delete the selected row when online.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: boolean deleteItem(long Index)</li> <li>• C++: BOOL &lt;CiCollectionName&gt;_DeleteItem (HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, PSI32 Index)</li> <li>• COM: Function DeleteItem(index As Long) As Boolean</li> </ul>

Data Collection Method	Action, Usage
Item(Index)	<p>Takes an item number as a parameter and returns a definition of the type that is stored in the specified row in the collection. For example, if the collection is a data collection, the return value is a DataRow. If the return value is a PropertyInfoCollection, then the return value is a PropertyInfo definition, and so on.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CiCollectionName&gt; item(long Index)</li> <li>• C++: HPSAPI_&lt;CI_COLLECTION_ITEM&gt; &lt;CiCollectionName&gt;_Item (HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, PSI32 Index) (HPSAPI_COMPINTFCPROPERTYINFOCOLLECTION, PSI32)</li> <li>• COM: Function Item(item As Long) As &lt;CI_COLLECTION_ITEM&gt;</li> </ul>
ItemByKeys(keys)	<p>Identifies and finds a specific item, based on keys. The keys vary according to the design of the collection.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CiCollectionName&gt; itemByKeys(String Key1, String Key2, ...)</li> <li>• C++: HPSAPI_&lt;CI_COLLECTION_ITEM&gt; &lt;CiCollectionName&gt;_ItemByKeys (HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, LPTSTR Key1, LPTSTR Key2, ...)</li> <li>• COM: Function ItemByKeys(KEY_1 As String, KEY_2,...) As &lt;CI_COLLECTION_ITEM&gt;</li> </ul>
CurrentItem	<p>Returns the current effective DataRow in the collection. The behavior is consistent with effective date rules that are used online. This method works with effective-dated records only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CiCollectionName&gt;currentItem()</li> <li>• C++: HPSAPI_&lt;CI_COLLECTION_ITEM&gt; &lt;CiCollectionName&gt;_CurrentItem(HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol)</li> <li>• COM: Function CurrentItem() As &lt;CI_COLLECTION_ITEM&gt;</li> </ul>

Data Collection Method	Action, Usage
<p>CurrentItemNum (CurrentItemNumber)</p>	<p>Returns the item number of the current effective DataRow in the collection. The behavior is consistent with effective date rules that are used online. This method works with effective-dated records only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: long currentItemNum()</li> <li>• C++: PSI32 &lt;CiCollectionName&gt;__CurrentItemNum (HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol)</li> <li>• COM: Function CurrentItemNum() As Long</li> </ul>
<p>GetEffectiveItem(DateString, SeqNum)</p>	<p>Returns the DataRow that would be effective for the specified date and sequence number. This is a more general case of the GetCurrentItem function, which returns the definition that is effective at this moment. This method works with effective-dated records only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: &lt;CiCollectionName&gt; getEffectiveItem(String Date, long SeqNum)</li> <li>• C++: HPSAPI_&lt;CI_COLLECTION_ITEM&gt; &lt;CiCollectionName&gt;_GetEffectiveItem(HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, LPTSTR Date, PSI32 SeqNum)</li> <li>• COM: Function GetEffectiveItem(Date As String, SeqNum As Long) As &lt;CI_COLLECTION_ITEM&gt;</li> </ul>
<p>GetEffectiveItemNum(DateString, SeqNum)</p>	<p>Returns the item number of the DataRow in the collection that would be effective for the specified date and sequence number. This is a more general case of the GetCurrentItemNum function, which returns the number of the definition that is effective at this moment. This method works with effective-dated records only.</p> <p>Use these interfaces to call with other programming languages.</p> <ul style="list-style-type: none"> <li>• Java: long getEffectiveItemNum(string Date, long SeqNum)</li> <li>• C++: &lt;CiCollectionName&gt;_GetEffectiveItemNum(HPSAPI_&lt;CI_COLLECTION_NAME&gt; hCol, LPTSTR Date, PSI32 SeqNum)</li> <li>• COM: Function GetEffectiveItemNum(Date As String, SeqNum As Long)</li> </ul>

## Enabling and Disabling Standard Methods

You can control whether standard methods are accessible at runtime.



Enabling standard methods

To enable or disable standard methods:

1. Select File, Definition Properties from the PeopleSoft Application Designer menu.

The Definition Properties dialog box appears.

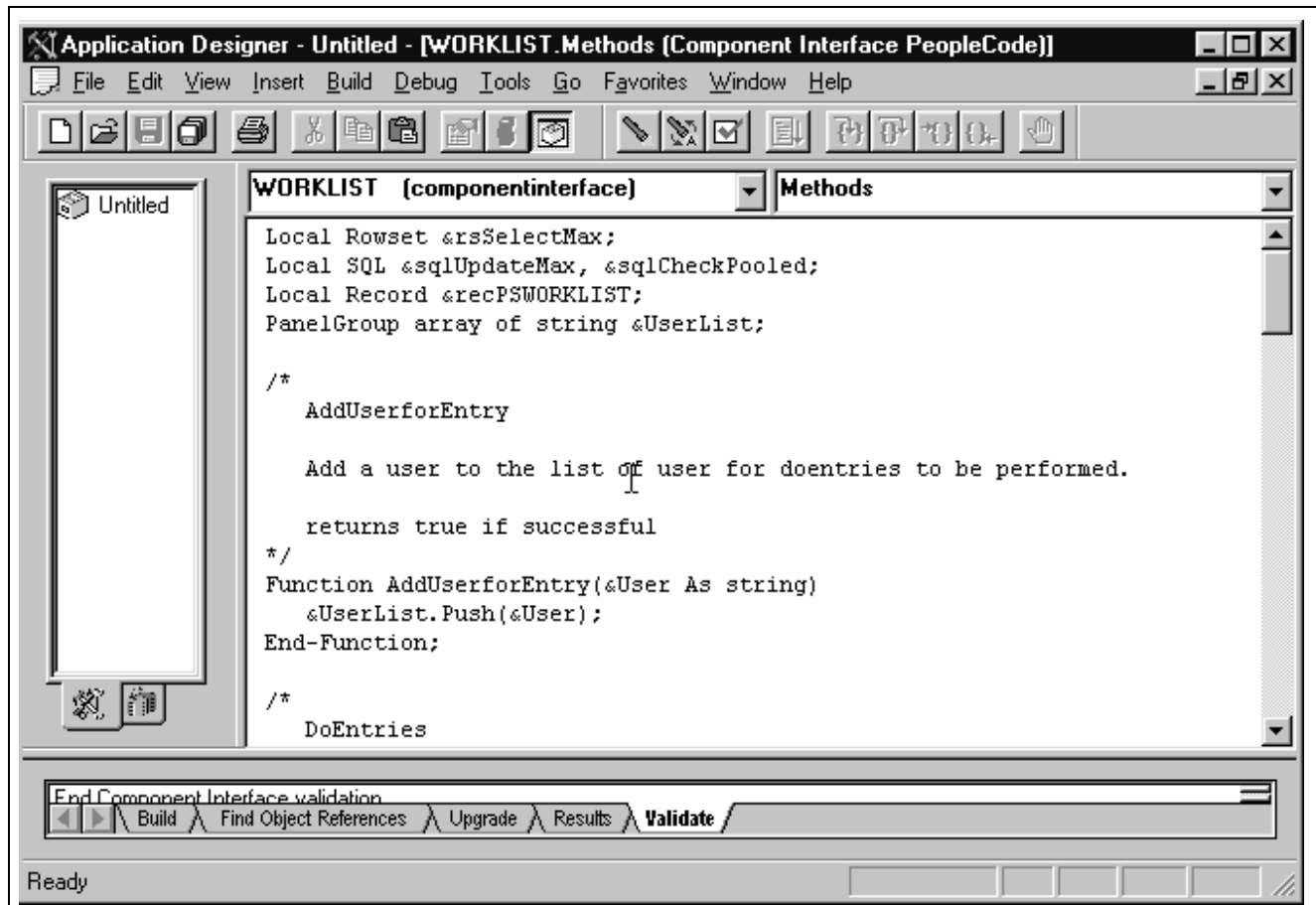
2. Select the Standard Methods tab.

You can enable or disable any of the standard methods selecting the corresponding check box. Doing so determines whether the method is available at runtime when the component interface is accessed.

The Create option is available only if the component interface has Create keys.

## Creating User-Defined Methods

This section discusses how to create user-define methods.



Creating user-defined methods in PeopleCode

To create a user-defined method:

1. Right-click anywhere in the component interface view.
2. Select View PeopleCode from the pop-up menu.

The PeopleCode editor appears. If you are using a new component interface, no PeopleCode will appear in the editor because no user-defined methods have been created.

3. Write the required PeopleCode functions.

PeopleCode functions that you write are stored in a single PeopleCode program that is attached to the component interface and associated with the Methods event.

---

**Note.** New user-defined methods do not appear in the list of methods until you save the component interface. Double-click the icon of any existing user-defined method to return to this PeopleCode program.

---

4. Set permissions for the methods that you created.

You must set permissions for every user-defined method. If you set permission to Full Access, at runtime that function is exposed to external systems as a method on the component interface object.

---

## Validating the Component Interface

Validation ensures that a component interface's structure is still valid. Over time, the structure of a component interface can become invalid due to component structural changes and modifications. For example, this can happen whenever a component deletes or adds a record or field. It can also happen if the keys on the component are added or removed. Properties and keys that no longer synchronize with their associated components are marked with an X icon.

---

**Note.** The validation process only determines whether the underlying component of a component interface has changed. It does not validate the PeopleCode that is associated with a component interface. To validate the PeopleCode, open the component and select Tools, Validate from the PeopleSoft Application Designer menu.

---

To correct an invalid component interface, you might have to delete properties for which there are no longer appropriate fields or records. If the structure of the source component has changed, you might have to delete old properties and re-add the new properties in their appropriate locations. You may also need to rename a property or collection.

To validate a component interface:

1. Open the component interface in PeopleSoft Application Designer.  
Validation occurs automatically whenever you open a component interface in PeopleSoft Application Designer.
2. Select Tools, Validate for Consistency from the PeopleSoft Application Designer menu to validate an open component interface.

As you change components or other related definitions, you should validate a component interface that is already open in PeopleSoft Application Designer.

---

**Note.** PeopleSoft Application Designer also validates each component interface upon its creation.

---

---

## Setting Component Interface Security

After creating a component interface, you must set security for it. Each individual method also needs to be provided security. Security for the component interface is provided through the PeopleSoft Internet Architecture pages. Component interface permissions are set at the permission list level in PeopleSoft security.

**Component Interface Permissions**

WF\_TIMEOUT\_DATA

Method	*Method Access
Cancel	Full Access
Find	Full Access
Get	Full Access
Save	Full Access

Full Access (All)

No Access (All)

OK Cancel

Setting access permissions for methods

To set up component interface security:

1. Log in to PIA through the browser, and select PeopleTools, Security, Permissions & Roles, Permission Lists.
2. Select the permission list for which you want to set security.

The Permission List component appears.

3. Access the Component Interfaces page.
4. Select the component interface for which you want to set security.

To add another component interface to the list, click the Add button.

5. Click Edit.

The Component Interface Permissions page appears, showing all of the methods (both standard and user-defined) in the component interface and their method access.

6. Set the access permission for each method.

Select Full Access or No Access. You must grant full access to at least one method to make the component interface available for testing and other online use.

7. Click OK when you're done.
8. Save the page.

## Testing the Component Interface

After setting the security for a component interface, you can test the contents and behavior using the component interface tester. You should test the component interface before using it in your external system. This proactive tool helps you discover problems with the underlying component or the component interface itself, including user defined methods. When you are testing a component interface, real data from the database is used. Therefore, if you save the information that you change by calling the Save method, the information is changed in the database.

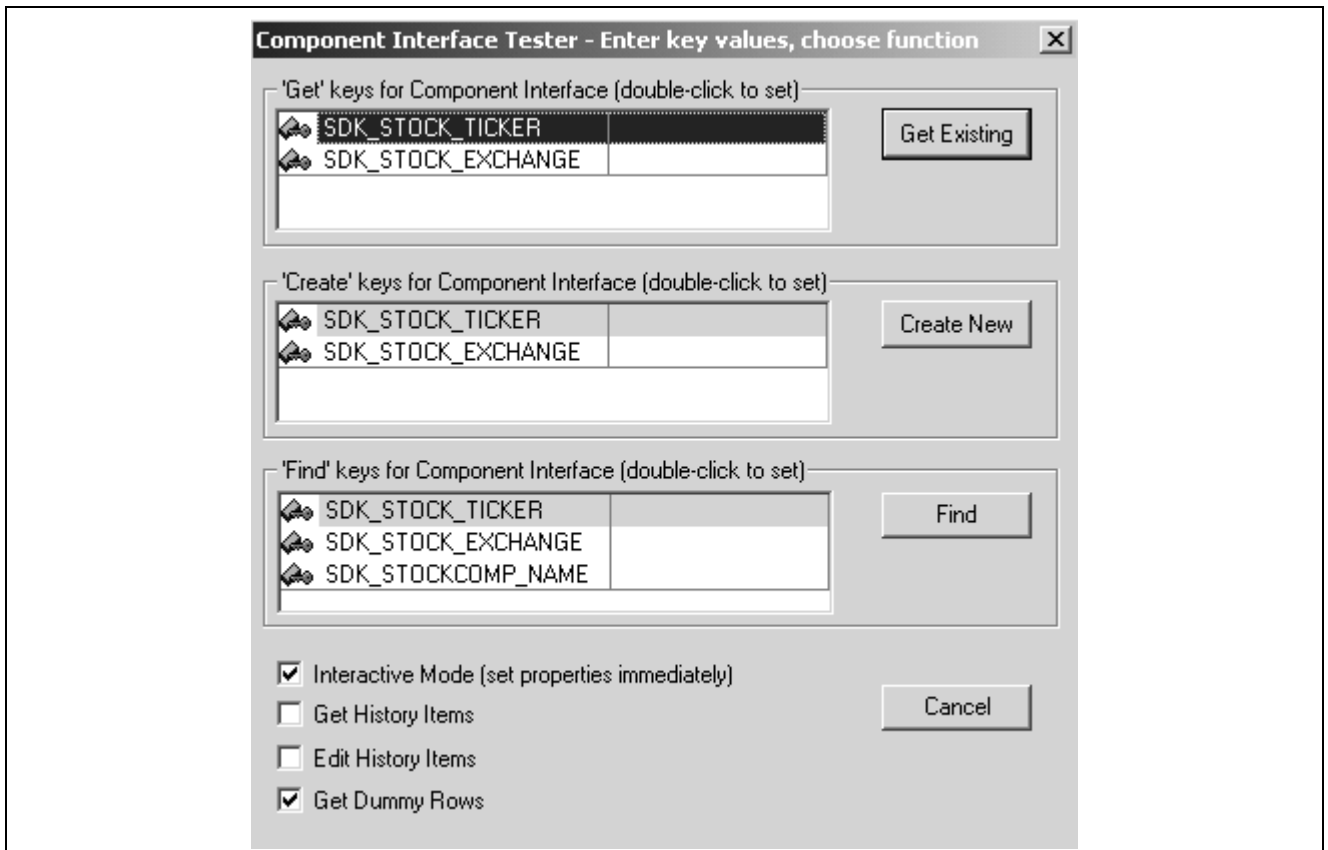
With the component interface tester, you can:

- Test the component interface in interactive mode.
- Retrieve history items.
- Test the standard, custom, and collection methods.

## Searching for a Component Interface to Test

To test the component interface, you search for the component interface to test, then you test it.

Access the Component Interface Tester search dialog box:



Component Interface Tester search dialog box

To search for a component interface to test:

1. Open the component interface in PeopleSoft Application Designer.
2. Select Tools, Test Component Interface from the PeopleSoft Application Designer menu.

The Component Interface Tester search dialog box appears. This dialog box displays the keys (in the left-hand columns) for getting, creating, or finding an instance of the component interface. The right-hand columns provide a place for you to enter sample key values for testing.

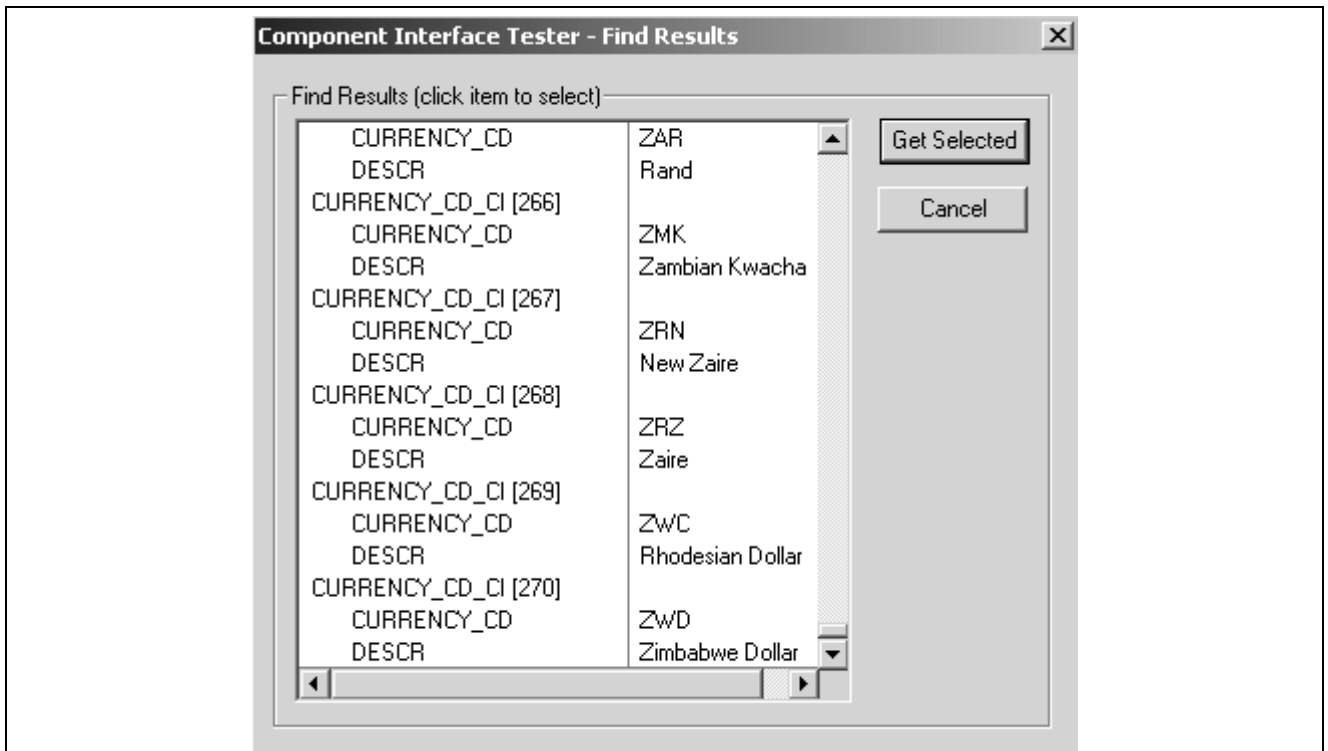
3. Enter key values.
  - a. Double-click the column to the right of any displayed keys.
  - b. Enter the value in the right-hand column.

The data that is used for the test corresponds to the key values that you enter here. In the preceding example, we've entered an employee ID of 6602.

<b>Interactive Mode</b>	<p>In interactive mode any action request occurs immediately. Each property being set causes an immediate trip to the application server (or database server in two-tier mode). This differs from noninteractive mode, in which actions are often held and later sent in batches. For example, in noninteractive mode, if you set a property, the property is not validated until you perform the save. However, in interactive mode the property is validated immediately. This means that edit processing (and other processing, such as FieldChange PeopleCode) occurs for each set property.</p> <p>Whether you select this option depends on how you expect a particular component interface to be used and what you are currently testing. In a real production system, this parameter can significantly affect performance, but it makes little difference in the test component. In noninteractive mode, errors and properties are not updated until a method is run. By default, Interactive Mode is selected in the component interface tester.</p>
<b>Get History Items</b>	<p>Selecting Get History Items retrieves history data. This option applies to effective-dated fields only and is equivalent to running in either Update/Display or Correction mode.</p>
<b>Edit History Items</b>	<p>Selecting Edit History Items enables editing and saving of history data. This option applies to effective-dated fields only and is equivalent to running in either Update/Display or Correction mode.</p>
<b>Get Dummy Rows</b>	<p>Specify whether to get dummy rows. This option is selected as a default.</p> <p>The component processor provides dummy rows to enable quick data entry when the level you are accessing does not have any data. Because of this an API that does not need this row finds it and ends up exposing it to the user. The application that uses the API now has to determine if the row is a dummy row or not and accordingly decides to execute Item or InsertItem.</p> <p>Setting the GetDummyRow to false enables the component interface processor to handle the counts accordingly. With this property set to false users do not have to use item and InsertItem when adding new data at levels 1 to 3. Instead they can comfortably always use InsertItem.</p>
<b>Get Existing</b>	<p>Clicking Get Existing is equivalent to opening a record in Update/Display or Correction mode online. It retrieves one instance from the database. After you click the Get Existing button, the Component Interface Tester dialog box appears.</p>
<b>Create New</b>	<p>Clicking Create New is equivalent to creating a new row in Add mode online. If your component does not support the Create method, this button is disabled. After you click the Create New button, the Component Interface Tester dialog box appears.</p>

### Getting Existing Records by Using Partial Keys

If you want to retrieve a partial key, click the Find button. The Find Results dialog box appears:

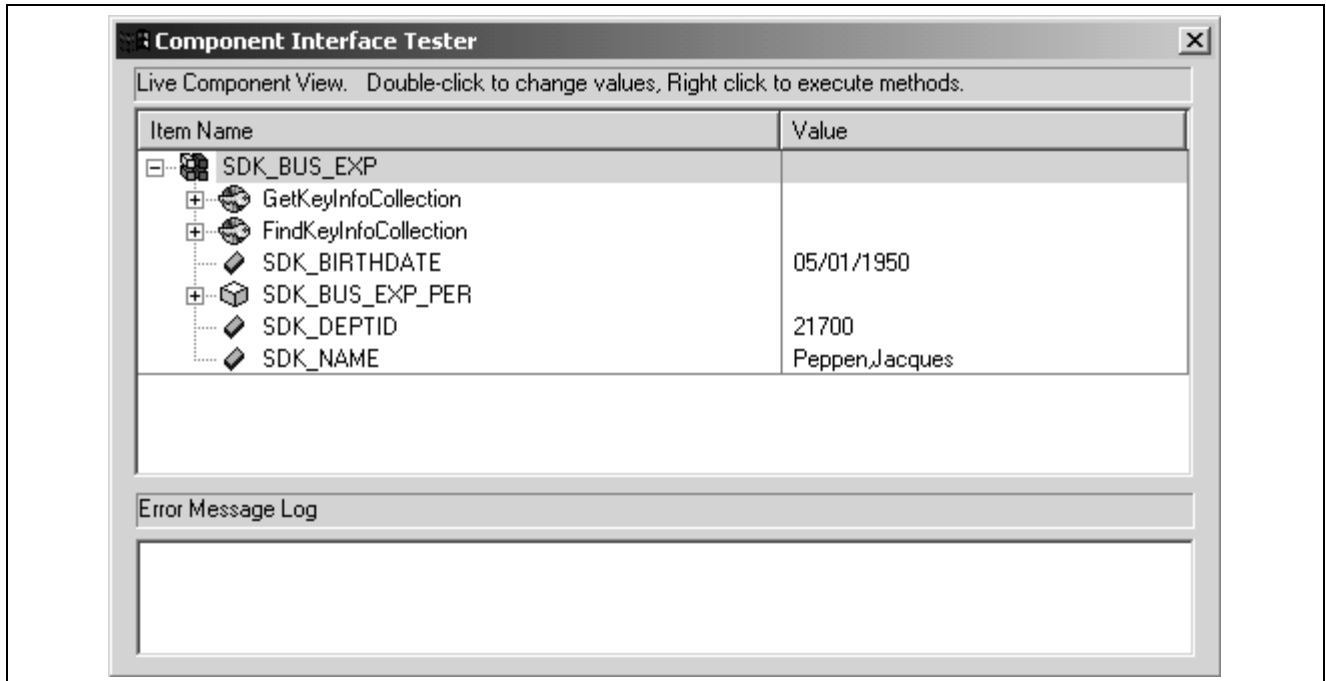


Component Interface Tester — Find Results dialog box

You then can choose the specific instance by selecting and clicking the Get Selected button. If you do not enter a partial key before clicking Find, all key values in the database are returned (subject to the maximum count of 300, just as when online). This is the same as calling the Find method through the component interface API, followed by selecting a value from the Find results, setting the Get key, and calling the Get method. After you click the Get Selected button, the Component Interface Tester dialog box appears.

## Testing the Component Interface

After you have searched for and retrieved the component interface, the Component Interface Tester dialog box appears.



Component Interface Tester dialog box

## Testing the Component Interface Properties

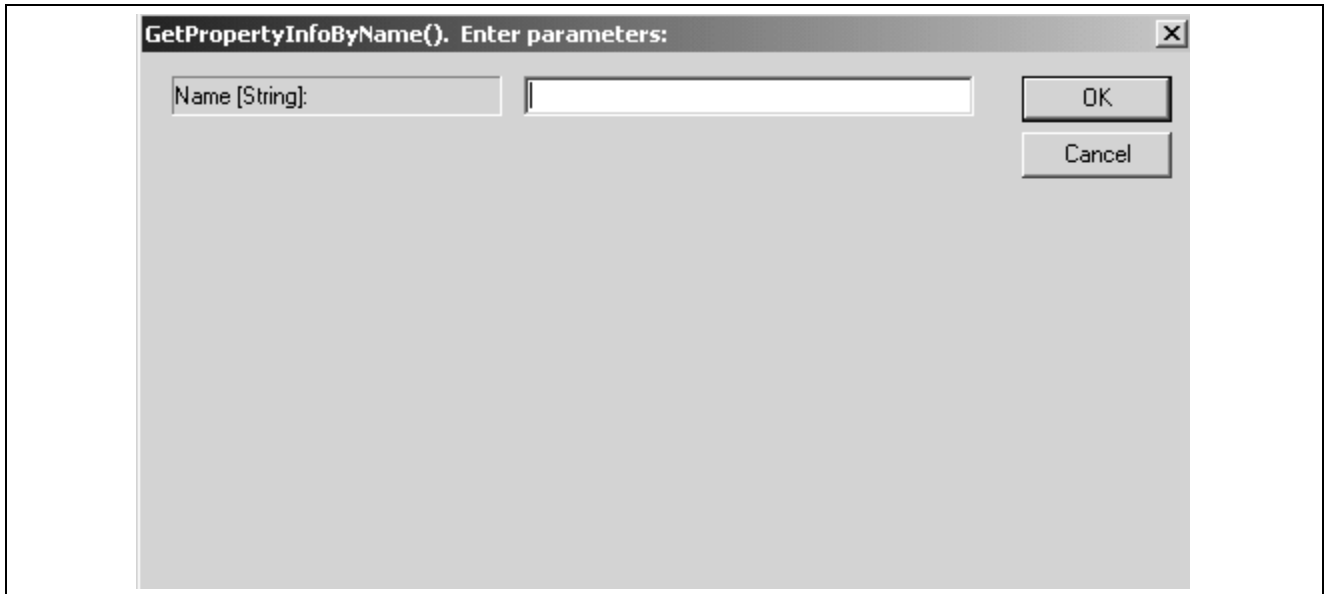
From the Component Interface Tester dialog box, change the value of a property, double-click a value, and enter a new value. Some basic validation is done when you leave the field, which is equivalent to exiting a field using the TAB key in the online case. This validation includes system edit, FieldChange PeopleCode events, and FieldEdit PeopleCode events. Further validation may be done when the Save method is called (SaveEdit, SavePreChange, Workflow, and SavePostChange). If errors or warnings are encountered, they are displayed in the Error Message Log area at the bottom of the window. The error message log displays the same text that would appear in the PSMessages collection of the Session object if you accessed the component through the Component Interface API.

## Testing Component Interface Methods

Test component interface methods by right-clicking the component interface name.

A pop-up menu appears showing the Save and Cancel standard methods and any user-defined methods that exist for the component interface. The Find, Create, and Get standard methods are not valid for an instantiated component, and therefore are not shown.

If a component interface method requires one or more parameters, a dialog box in which you can enter the parameters appears. After the method gets executed, the same dialog box appears again, displaying changes to the parameters that were caused by the method. The return value of the function appears in the title of the dialog box. If a component interface requires no parameters, you do not see the initial dialog box, but you do see the return value dialog box following the function call.



GetPropertyInfoByName(). Enter parameters: dialog box

**Note.** Because running a component interface method can result in a change to the component interface structure, PeopleSoft Application Designer always redraws the component interface tree in its collapsed form following a method call.

## Testing Collection Methods

Test collection methods by right-clicking the collection name.

A pop-up menu appears, showing the standard collection methods. Select the collection method that you want to test for this component interface. After you select a collection method to test, the Enter parameters dialog box prompts you to enter an item number for the collection method that you are testing. The value that you enter for index [Number] is used to retrieve, insert, or delete an item, according to the following rules.

After you enter an index number, the result appears in the dialog box. If there is a return value, it is displayed in the title bar. Otherwise the message No value is displayed. Click OK or Cancel to dismiss the dialog box.

## Collection Method Rules

This table contains the collection method rules:

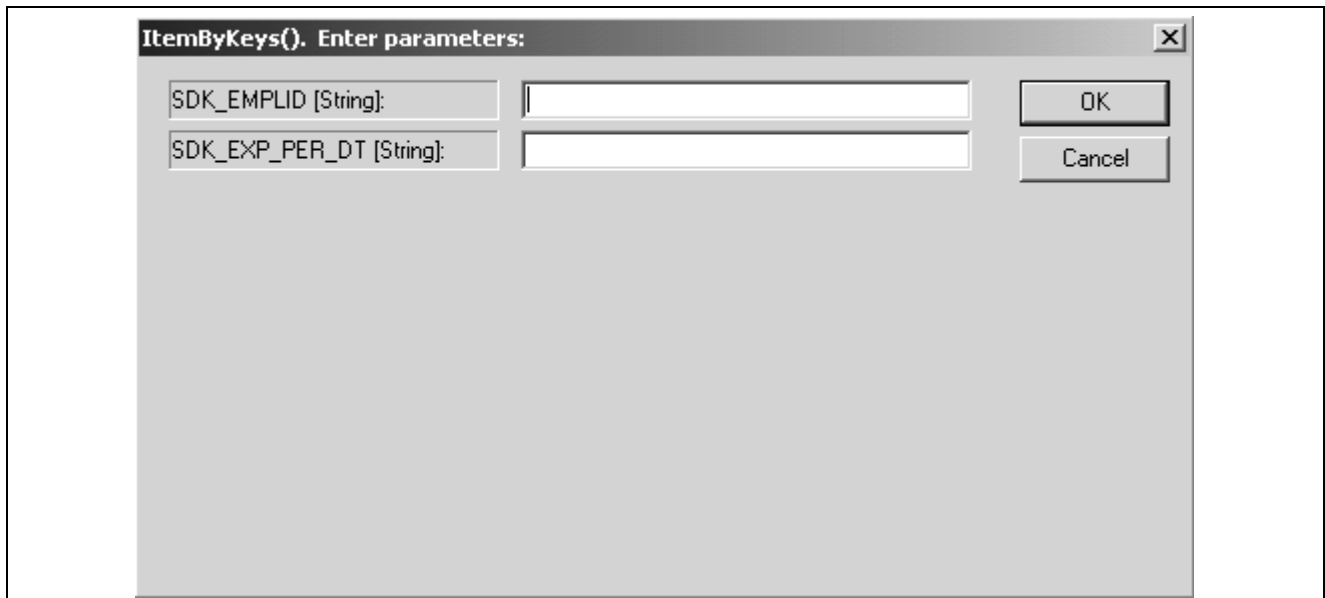
Collection Method	Purpose
Item(index)	Returns the row at the specified index. Only the success or failure of this routine is of interest from inside the test component.
InsertItem(index)	Inserts a new row either before the index that you specify if the collection is effective-dated or following the index if it isn't effective-dated.
DeleteItem(index)	Deletes the row that is designated by the index number that you specified in the Enter parameters dialog box.

Collection Method	Purpose
ItemByKeys(key1, key2, ...)	Returns the row corresponding to the specified keys. Only the success or failure of this routine is of interest from inside the test component.
CurrentItem	This method returns the effective row in an effective-dated record. Only the success or failure of this routine is of interest from inside the test component.
GetEffectiveItem(DateString, SeqNum)	Returns the DataRow that would be effective for the specified date and sequence number. This is a more general case of the GetCurrentItem function, which returns the definition that is effective at this moment. This method works with effective-dated records only.
GetEffectiveItemNum(DateString, SeqNum)	Returns the item number inside the collection of the DataRow that would be effective for the specified date and sequence number. This is a more general case of the GetCurrentItemNum function, which returns the number of the definition that is effective at this moment. This method works with effective-dated records only.

**Note.** Component Interface Classes contains information about collection methods.

## Determining ItemByKeys Parameters

You can get the signature for the ItemByKeys method (or any other method) when testing a component interface. This is helpful for the ItemByKeys method, because its signature is different for each component interface.



Viewing the signature of the ItemByKeys method

To determine ItemByKeys parameters:

1. Open the definition.
2. Select Tools, Test Component.
3. Find or get an appropriate populated component.
4. Navigate to the appropriate collection.
5. Right-click, and select ItemByKeys from the pop-up menu.

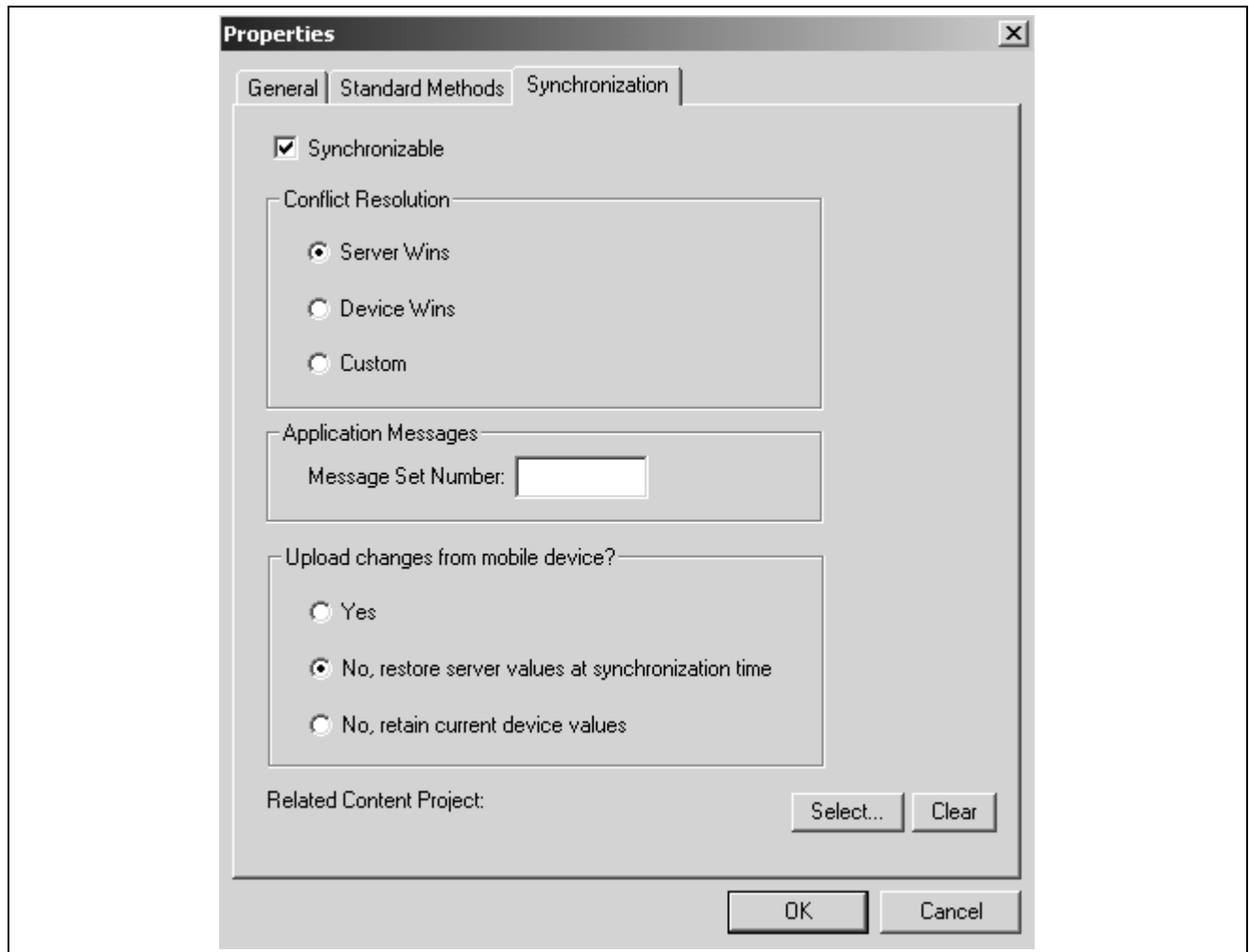
A dialog box appears, showing the specific parameters and types and the order in which you should call ItemByKeys.

In the preceding example, the keys for the SDK\_BUS\_EXP\_PER ItemByKeys method are SDK\_EMPID (String) and SDK\_EXP\_PER\_DT (String).

---

## Understanding Synchronization

The Component Interface Properties Synchronization tab is used with PeopleSoft Mobile Agent. PeopleSoft Mobile Agent extends the functionality of PeopleSoft Pure Internet Architecture to disconnected mobile devices, allowing users to continue working with their PeopleSoft applications on a laptop computer or personal digital assistant (PDA) while disconnected from the internet or local network.



Properties—Synchronization tab

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Mobile Agent*, “Using Synchronizable Component Interfaces”.

## Writing a Component Interface Program

The following chapters in this PeopleBook describe how to write component interface programs in several programming languages.

Also, the PeopleTools PeopleCode Reference contains a chapter that describes the component interface classes, including detailed instructions on the life cycle of a component interface and how to implement a component interface program in PeopleCode. You can use this information to help design your component interface program in other programming languages.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference*, “Component Interface Classes”.

---

## Runtime Considerations

In many ways, accessing a component interface is functionally equivalent to working with an online component. However, there are some important differences between component interfaces and components. This section describes how those differences affect interactive operation, functionality designed for graphical interfaces, client versus server operation, and several miscellaneous situations. These considerations, unless otherwise noted, apply to all the programming languages listed in this manual.

### General Considerations

This section discusses general considerations for component interface programs.

#### WinMessage Unavailable

You can't use WinMessage in a component that will be used to build a component interface. Use MsgGet() instead.

#### Email from a Component Interface

To use a component interface to send email, use TriggerBusinessEvent PeopleCode event, not SendMail.

#### Related Display

Related display fields are not available for use in a component interface as they are not held in the buffer context that the component interface uses.

#### Row Inserts

If RowInserts have been disabled for a page, you must take care when calling inserts against the corresponding component interface. Any PeopleCode associated with push buttons used on the page to add rows will not be invoked by the component interface when an insert is done.

---

**Note.** If a component has logic that inserts rows on a RowInsert, the component interface cannot identify the change and locate the rows that were inserted by the application code. Generic interfaces such as Excel to Component Interfaces utility and the WSDLToCI will not function correctly when using this type of dynamic insert.

---

## Scope Conflicts

This section discusses scope conflicts for component interface programs.

### Infinite Processing Loops

A component interface should not call itself in any of the PeopleCode included within its component definition, because this may result in an infinite loop of the component interface. A component interface also should not call itself from a user-defined method.

### Multiple Instances of a Component Interface

Because of potential memory conflicts, COM or C++ applications shouldn't create multiple, simultaneous instances of the same component interface, either within a single procedure, or in both a "parent" and a "child" procedure.

## Interactive Mode

This section discusses interactive mode considerations for component interface programs.

### UNIX Server Performance

If you're using a component interface as part of a batch process in which thousands of rows are being inserted, running in interactive mode may reduce performance enough on some UNIX servers to produce a connection failure. Prevent this by setting the InteractiveMode property to False.

### Hidden Edit Validation Errors

If the InteractiveMode property is set to True, and if a transaction sets a property to a value that isn't allowed in a prompt edit field, the edit field value is reset back to its original value. The error is logged in the PSMessages collection; however, the Save method runs without errors. Check the value of both the Save method and the collection ErrorPending property to discover all of the errors.

## CHAPTER 4

# Programming Component Interfaces in Java

This chapter discusses how to:

- Build the component interface APIs.
- Set up the Java environment.
- Generate a Java runtime code template.
- Use and understand the generated Java code.

---

## Building APIs in Java

If you plan to access your component interface from a Java external application, you must create a component interface API. The APIs are in the form of \*.java source code files, which should be compiled into Java classes.

To build the component interface bindings:

1. Open any component interface definition in PeopleSoft Application Designer.

Use any component interface definition, because you can build APIs for all of them, regardless of which one is open.

2. Select Build, PeopleSoft APIs.

The Build PeopleSoft API Bindings dialog box appears.

3. Select the Build check box in the Java Classes group box.

For the target directory, enter the directory in which you want the Java class source files to be created. This directory is typically <PS\_HOME>\class.

4. Click OK to build the bindings that you selected.

The files that constitute the bindings are built in the location that you specified. If the operation was successful, a Done message appears in the PeopleSoft Application Designer Build window.

5. Compile the APIs that you've just generated.

You could use these commands:

```
cd %PS_HOME%\class\PeopleSoft\Generated\CompIntfc
javac -classpath %PS_HOME%\class\psjoa.jar *.java
cd c:\pt8\class\PeopleSoft\Generated\PeopleSoft
javac -classpath %PS_HOME%\class\psjoa.jar *.java
```

---

## Setting Up the Java Environment

When deploying component interfaces on a local client machine or web server with Java bindings, you must have:

- The third party Java application.
- The PeopleSoft application server and database.
- The Java Virtual Machine (JVM) supplied with Sun Microsystems' Java Development Kit (JDK) found in the %PS\_HOME%\JRE directory.

To set up your client machine to access the component interface API using Java:

1. Install Sun Microsystems' JDK to enable the JVM, if it is not already installed.  
Sun Microsystems' Java Development Kit (JDK) can be found in the %PS\_HOME%\JRE directory
2. Set the environment variable PATH to include the directory containing jvm.dll.  
For example, c:\bea\jdk131\jre\bin\classic; or, if the PeopleTools install is done locally the path is <PS\_HOME>\jre\bin\hotspot.
3. Set the environment variable CLASSPATH to include:
  - a. The file psjoa.jar (typically <PS\_HOME>\class\psjoa.jar).
  - b. The target directory selected during the Build API process (<PS\_HOME>\class).

---

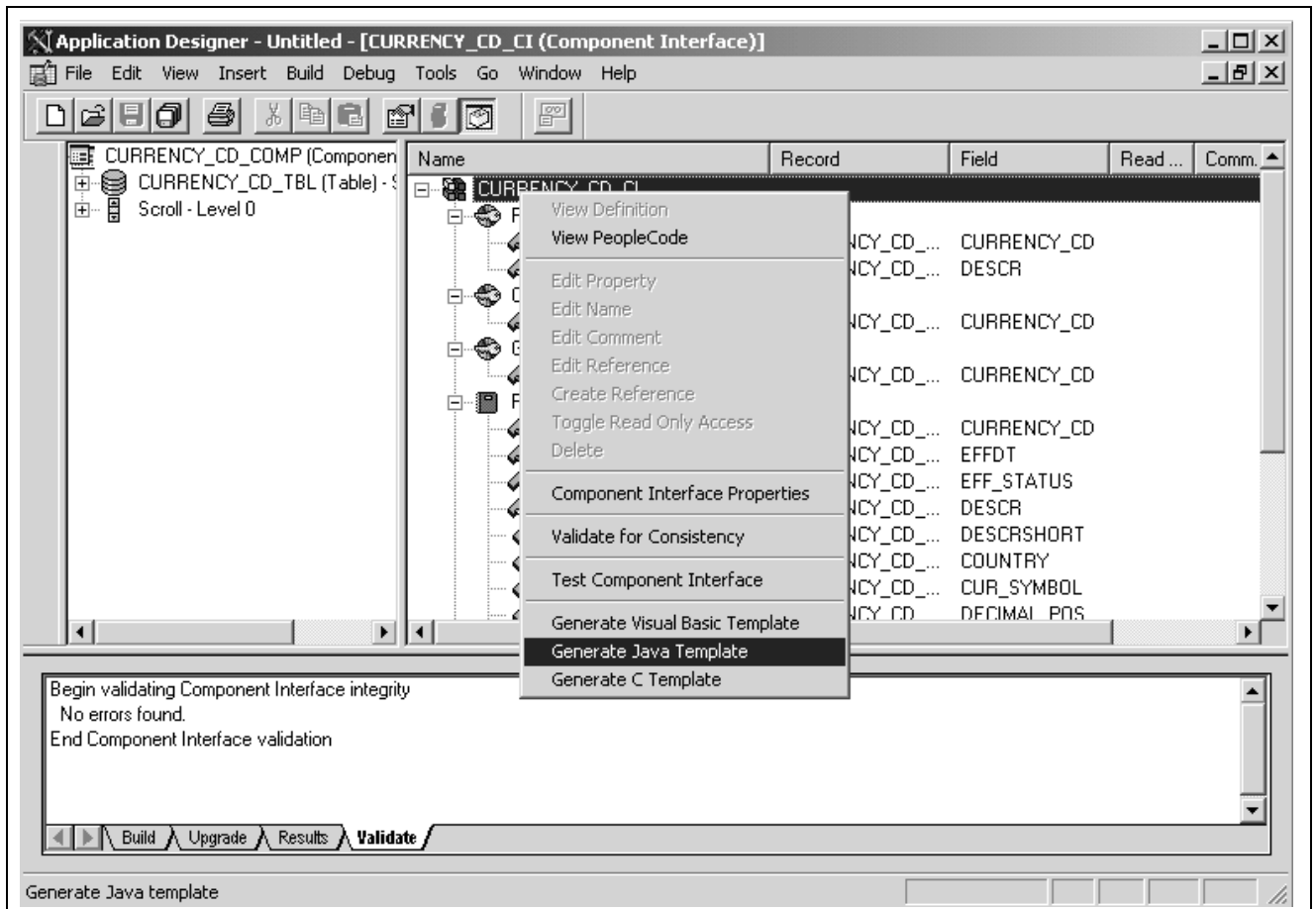
**Note.** In previous releases, sites using UNIX servers received the following error when invoking a component interface through the PeopleSoft Java Object Adapter (PSJOA): *PSPProperties not loaded from file*. To resolve this issue, copy the pstools.properties file to the Component Interface execution directory.

---

---

## Generating a Java Runtime Code Template

To access a component interface through PeopleSoft APIs using Java, PeopleSoft Application Designer generates a template in the form of boilerplate Java code that you can adapt to your purposes. This section describes how to generate the template code.



#### Generating Java template

To generate a Java template for a component interface:

1. Open a component interface definition in Application Designer.
2. Right-click anywhere in the definition view to display the pop-up menu.
3. Select Generate Java Template.

When the template is successfully generated, a message appears stating the name and location of the template file.

---

**Note.** The template file is generated in the directory specified by the TEMP or TMP system environment variable on your client machine.

---

4. Edit the generated file and modify the source code to suit your needs.
5. Compile the source code to generate a *class* file.

In the case of the example used in this manual, you could use this command:

```
javac -classpath c:\temp;c:\pt8\class;c:\PT8\class\psjoa.jar SDK_BUS_EXP.java
```

---

## Understanding the Java Template

You can use the Java template as a starting point for your Java program. This section contains a skeleton of the generated Java template for a component interface named SDK\_BUS\_EXP, which is part of the component interface SDK. The template has been edited for length.

Import all the required classes.

```
import java.io.*;
import psft.pt8.joa.*;
import PeopleSoft.Generated.CompIntfc.*;
public class SDK_BUS_EXP {
    public static ISession oSession;
    .....
    public static void main (String args[]) {
        try {
            //***** Set Connect Parameters *****
            String strServerName, strServerPort, strAppServerPath;
            String strUserID, strPassword;
            .....
            //Build Application Server Path
            strAppServerPath = strServerName + ":" + strServerPort;
```

---

**Note.** To enable Jolt failover and load balancing in the PeopleSoft Internet Architecture, you can supply multiple application server domains for the strAppServerPath variable. Separate the domain names with a comma, and make sure there are no spaces. For example: strAppServerPath = //APPSRVR1:8000,//APPSRVR2:9000

---

Create the PeopleSoft Session object to enable access to the PeopleSoft system.

The Session object controls the environment and allows you to do error handling for all APIs from a central location.

```
//***** Create PeopleSoft Session Object *****
oSession = API.createSession();
```

Connect to the PeopleSoft application server by using the connect method of the Session object.

```
//***** Connect to the App Server *****
if (!oSession.connect(1, strAppServerPath, strUserID,
    strPassword, null)) {
    System.out.println("\nUnable to Connect to Application Server.");
    ErrorHandler();
    return;
}
```

Get a reference to the component interface providing its name. (A runtime error occurs if the component interface does not exist.)

```
ISdkBusExp oSdkBusExp;
String ciName;
ciName = "SDK_BUS_EXP";
oSdkBusExp = (ISdkBusExp) oSession.getCompIntfc(ciName);
if (oSdkBusExp == null) {
    System.out.println("\nUnable to Get Component Interface " +
```

```

        ciName);
    ErrorHandler();
    return;
}

//***** Set the Component Interface Mode *****
oSdkBusExp.setInteractiveMode(false);
oSdkBusExp.setGetHistoryItems(true);
oSdkBusExp.setEditHistoryItems(false);

```

Set the keys for the component interface. In this example SDK\_EMPLID is the Get key.

```

//***** Set Component Interface Get/Create Keys *****
String strSdkEmplid;
System.out.print("\nEnter SdkEmplid: ");
strSdkEmplid = inData.readLine();
oSdkBusExp.setSdkEmplid(strSdkEmplid);

```

The get() method retrieves data from the database, associated with the key values.

```

//***** Execute Get *****
if (!oSdkBusExp.get()) {
    System.out.println("\nNo rows exist for the specified keys.\nFailed to⇒
get the Component Interface.");
    ErrorHandler();
    return;
}
.....

```

Get and print properties at level 0.

```

System.out.println("oSdkBusExp.SdkName: " +
    oSdkBusExp.getSdkName());
.....

```

Similar code is generated for the properties SDK\_BIRTHDATE and SDK\_DEPTID.

Get collection at level 1 (SDK\_BUS\_EXP\_PER).

```

ISdkBusExpSdkBusExpPerCollection oSdkBusExpPerCollection;
ISdkBusExpSdkBusExpPer oSdkBusExpPer;
oSdkBusExpPerCollection = oSdkBusExp.getSdkBusExpPer();

```

Get and print properties at level 1.

```

for (int i17 = 0;
    i17 < oSdkBusExpPerCollection.getCount(); i17++) {
    oSdkBusExpPer = oSdkBusExpPerCollection.item(i17);

    System.out.println("oSdkBusExpPer.SdkExpPerDt: " +
        oSdkBusExpPer.getSdkExpPerDt());
    .....
}

```

Similar code is generated for the properties SDK\_EMPLID and SDK\_BUS\_EXP\_SUM in the SDK\_BUS\_EXP\_PER collection.

Get collection at level 2 (SDK\_BUS\_EXP\_DTL).

```

ISdkBusExpSdkBusExpPerSdkBusExpDtlCollection

```

```

        oSdkBusExpDtlCollection;
        ISdkBusExpSdkBusExpPerSdkBusExpDtl oSdkBusExpDtl;
        oSdkBusExpDtlCollection = oSdkBusExpPer.getSdkBusExpDtl();

```

Get and print properties at level 2.

```

    for (int i211 = 0;
         i211 < oSdkBusExpDtlCollection.getCount(); i211++) {
        oSdkBusExpDtl = oSdkBusExpDtlCollection.item(i211);

        System.out.println("oSdkBusExpDtl.SdkChargeDt: " +
            oSdkBusExpDtl.getSdkChargeDt());
        .....

```

Similar code is generated for the properties SDK\_EMPID, SDK\_EXP\_PER\_DT, SDK\_EXPENSE\_CD, SDK\_EXPENSE\_AMT, SDK\_CURRENCY\_CD, SDK\_BUS\_PURPOSE, and SDK\_DEPTID.

```

    }
}

```

Disconnect from the PeopleSoft application server by using the disconnect method of the Session object. This method clears the buffers and releases the memory.

```

//***** Disconnect from the App Server *****
    oSession.disconnect();
    return;
}
catch (Exception e) {
    e.printStackTrace();
    System.out.println("An error occurred: ");
    ErrorHandler();
}
}
}
}

```

## CHAPTER 5

# Programming Component Interfaces in C++

This chapter discusses how to:

- Build the component interface APIs.
- Set up the C++ environment.
- Generate a C++ runtime code template.
- Use and understand the generated C++ code.

---

## Building APIs for C++

If you plan to access your component interface from a Java external application, you must create a component interface API. The APIs are in the form of C header files (\*.h), which need to be included in the calling program.

To build the component interface bindings:

1. Open any component interface definition in PeopleSoft Application Designer.

Use any component interface definition, because you can build APIs for all of them, regardless of which one is open.

2. Select Build, PeopleSoft APIs.

The Build PeopleSoft API Bindings dialog box appears.

3. Select the Build check box in the C Header Files group box.

For the target directory, enter the directory in which you want the C++ header file to be created, typically <PS\_HOME>\bin\client\winX86

4. Click OK to build the bindings that you selected.

The peoplesoft\_peoplesoft.\_i.h file that constitutes the bindings is built in the location that you specified. If the operation was successful, a Done message appears in the PeopleSoft Application Designer Build window.

---

## Setting Up the C++ Environment

When deploying component interfaces on a local client machine with C++ bindings, you must have:

- The third-party C++ application.
- The PeopleSoft application server and database.

- The Java Virtual Machine (JVM) supplied with Sun Microsystems' Java Development Kit (JDK) found in the %PS\_HOME%\JRE directory
- Your compiler, configured for the C++ project.

### Third-Party Applications

For applications written in C or C++, note the following:

- The function names generated by the Build APIs process can be quite long. You may want to consider creating classes within your C++ code to mask this length throughout your program.
- When you create your installation for your C or C++ program, make sure you include the setup of the path to the psapiadapter.dll.

## Setting Up a Client Machine to Access a C++ API

To set up your client machine to access the component interface API using C++:

1. Install the PeopleSoft File Server.  
See *PeopleSoft Installation Guide*, "Using the PeopleSoft Installer."
2. Install the Java Virtual Machine (JVM) supplied with Sun Microsystems' Java Development Kit (JDK), found in the %PS\_HOME%\JRE directory.
3. Set the environment variable PATH to include the directory containing jvm.dll.  
For example, it's c:\bea\jdk131\jre\bin\classic; or, if the PeopleTools install is done locally the path is <PS\_HOME>\jre\bin\hotspot.
4. Set the environment variable CLASSPATH to include the psjoa.jar file (typically <PS\_HOME>\class\psjoa.jar).
5. Set the environment variable PS\_HOME to point to the installed PeopleSoft PeopleTools directory (for example, c:\pt840).

## Configuring a Compiler for the C++ Project

The following are general instructions tell how to configure a compiler for the C++ project.

To configure a compiler for the C++ project:

---

**Note.** These instructions assume you're using Microsoft Visual C++. If you use a different compiler, apply the equivalent settings for that product.

---

1. Create a new project in Microsoft Visual C++.
2. Select Tools, Options.
3. Select the Directories tab.
4. Click the New button in the Options dialog box.
5. Enter the path to the SDK include files.

For example:

```
C:\PT840\SDK\PSCOMPINTFC\SRC\C++\SAMPLES\INC
```

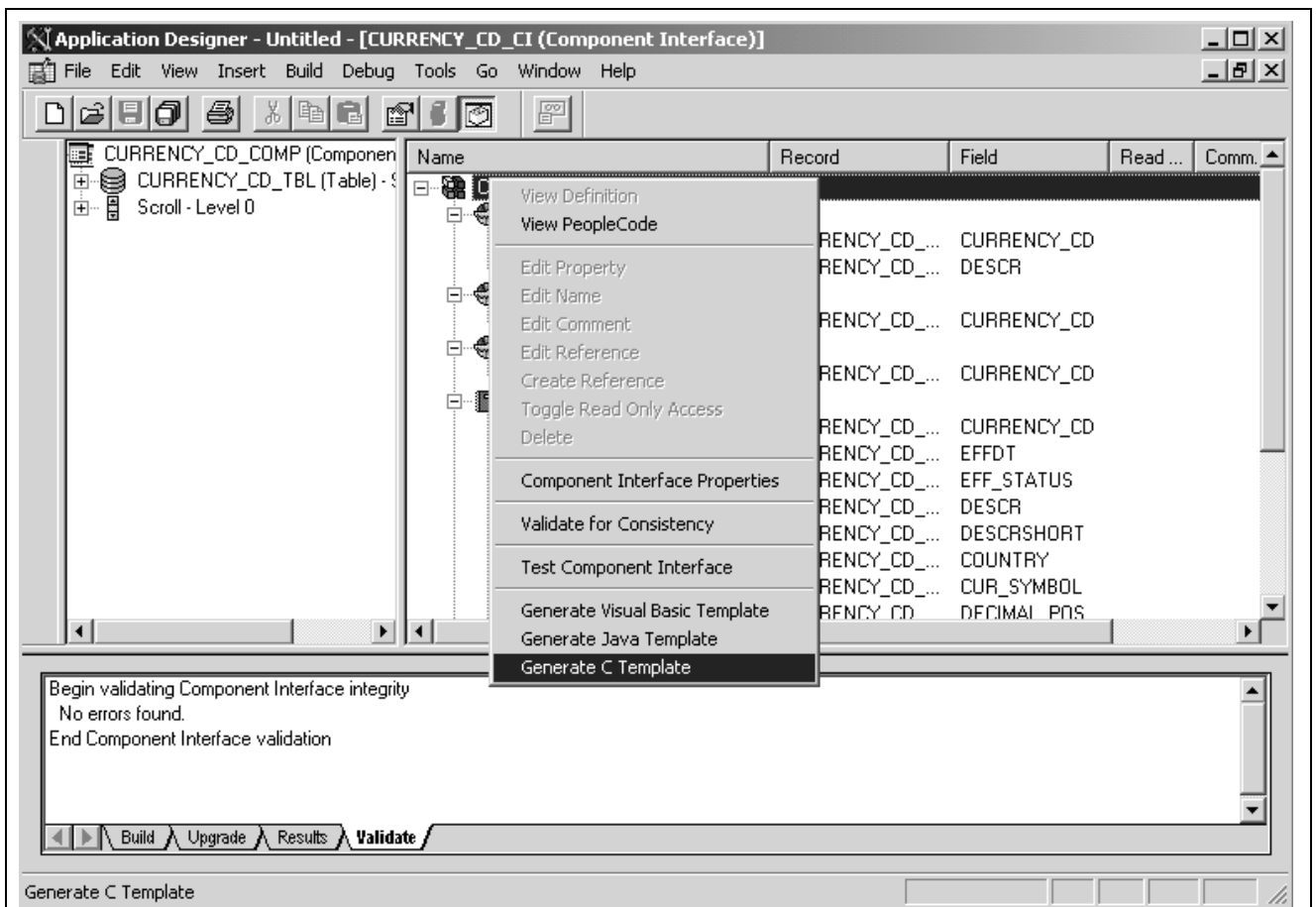
6. Click OK to save the options.

7. Open the Project Settings dialog box.
8. Select the C/C++ tab.
9. Select the General category.
10. Add PS\_WIN32 to the preprocessor definitions.
11. Select the Link tab.
12. Select the Input category.
13. Specify the full path to psapiadapter.lib for the Object/library modules.
 

This is typically <PS\_HOME>\src\lib\psapiadapter.lib. Make sure that this is the only entry for psapiadapter.lib.
14. Click OK to save the settings.

## Generating a C++ Runtime Code Template

To access a component interface through PeopleSoft APIs using C++, PeopleSoft Application Designer generates a template in the form of boilerplate C++ code that you can adapt to your purposes. This section describes how to generate the template code.



Generating C++ template

To generate a C++ template for a component interface:

1. Open a component interface definition in PeopleSoft Application Designer.
2. Right-click anywhere in the definition view to display the pop-up menu.
3. Select *Generate C Template*.

When the template is successfully generated, a message appears stating the name and location of the template file.

---

**Note.** The template file is generated in the directory specified by the TEMP or TMP system environment variable on your client machine.

---

4. Add the generated template file to the project.

In Microsoft Visual C++:

- a. Open the project created earlier.
  - b. Select Project, Add To Project, Files.
  - c. Select the generated file.
  - d. Click *OK*.
5. Edit the generated file and modify the source code to suit your needs.
  6. Build the project to generate an executable (.exe) file.

---

## Understanding the C++ Template

The C++ template can be used as a starting point for your C++ program. This section contains a skeleton of the generated C++ template for a component interface named SDK\_BUS\_EXP, which is part of the component interface SDK. The template has been edited for length.

Include all the required header files.

```

#ifdef PS_WIN32
#include "stdafx.h"
#endif

#include "cdef.h"
#include "apiadapterdef.h"
#include "PSApiAdapterInter.h"
#include "PSApiExternalLib.h"
#include "peoplesoft_peoplesoft_i.h"
#include <stdio.h>
#include <iostream.h>
#include <wchar.h>

HPSAPI_SESSION hSession;
TCHAR tmpValue[1024];

.....

```

```

void main(int argc, char* argv[])
{
    /****** Set Connect Parameters *****/
    TCHAR strServerName[40], strServerPort[10], strAppServerPath[80];
    TCHAR strUserID[80], strPassword[80];
    .....

    //Build Application Server Path
    _stprintf(strAppServerPath, _T("%s:%s"), strServerName, strServerPort);

```

---

**Note.** To enable Jolt failover and load balancing in the PeopleSoft Internet Architecture, you can supply multiple application server domains for the strAppServerPath variable. Separate the domain names with a comma, and make sure there are no spaces. For example: strAppServerPath = //APPSRVR1:8000,//APPSRVR2:9000

---

Create the PeopleSoft Session object to enable access to the PeopleSoft system.

The Session object controls the environment and allows you to perform error handling for all APIs from a central location.

```

    /****** Create PeopleSoft Session *****/
    PSAPIVARBLOB ExternalAuth;
    memset(&ExternalAuth, 0, sizeof(PSAPIVARBLOB));
    hSession = PSApiCreateSession();
    if (!hSession)
    {
        wprintf(L"\nUnable to Create Session\n");
        return;
    }

```

Connect to the PeopleSoft application server by using the Session\_Connect() function.

```

    /****** Connect to the App Server *****/
    if (!Session_Connect(hSession, 1, strAppServerPath, strUserID, strPassword, =>
    ExternalAuth))
    {
        wprintf(L"\nUnable to Connect to Application Server\n");
        ErrorHandler();
        return;
    }

```

Get a reference to the component interface providing its name. (A runtime error occurs if the component interface does not exist.)

```

    /****** Get Component Interface *****/
    HPSAPI_SDK_BUS_EXP hSdkBusExp;
    TCHAR ciName[30];
    _tcscpy(ciName, _T("SDK_BUS_EXP"));
    hSdkBusExp = (HPSAPI_SDK_BUS_EXP) Session_GetCompIntfc(hSession,
    ciName);
    if (!hSdkBusExp)
    {
        wprintf(L"\nUnable to Get Component Interface %s\n", ciName);
        ErrorHandler();
    }

```

```

    return;
}

//***** Set the Component Interface Mode *****
SdkBusExp_SetInteractiveMode(hSdkBusExp, false);
SdkBusExp_SetGetHistoryItems(hSdkBusExp, true);
SdkBusExp_SetEditHistoryItems(hSdkBusExp, false);

```

Set the keys for the component interface. In this example SDK\_EMPLID is the Get key.

```

//***** Set Component Interface Get/Create Keys *****
TCHAR strSdkEmplid[80];
wprintf(L"\nEnter SdkEmplid: ");
_getts(strSdkEmplid);
SdkBusExp_SetSdkEmplid(hSdkBusExp, strSdkEmplid);

```

The <CI\_NAME>\_Get() function retrieves data from the database associated with the key values.

```

//***** Execute Get *****
if (!SdkBusExp_Get(hSdkBusExp))
{
    wprintf(L"\nUnable to Get Component for the Search keys provided.\n");
    ErrorHandler();
    return;
}

```

Get and print properties at level 0.

```

wprintf(L"SdkBusExp.SdkName: %s\n",
    printProperty(SdkBusExp_GetSdkName(hSdkBusExp), tmpValue));

```

Similar code is generated for the properties SDK\_BIRTHDATE and SDK\_DEPTID.

Get collection at level 1 (SDK\_BUS\_EXP\_PER).

```

HPSAPI_SDK_BUS_EXP_SDK_BUS_EXP_PERCOLLECTION
    hSdkBusExpSdkBusExpPerCollection;
HPSAPI_SDK_BUS_EXP_SDK_BUS_EXP_PER hSdkBusExpSdkBusExpPer;
hSdkBusExpSdkBusExpPerCollection =
    SdkBusExp_GetSdkBusExpPer(hSdkBusExp);

```

Get and print properties at level 1.

```

for (int i17 = 0; i17 < SdkBusExpSdkBusExpPerCollection_GetCount
    (hSdkBusExpSdkBusExpPerCollection); i17++)
{
    hSdkBusExpSdkBusExpPer = SdkBusExpSdkBusExpPerCollection_Item
        (hSdkBusExpSdkBusExpPerCollection, i17);
    wprintf(L"oSdkBusExpSdkBusExpPer.SdkExpPerDt: %s\n",
        printProperty
        (SdkBusExpSdkBusExpPer_GetSdkExpPerDt(hSdkBusExpSdkBusExpPer),
            tmpValue));
}

```

Similar code is generated for the properties SDK\_EMPLID and SDK\_BUS\_EXP\_SUM in the SDK\_BUS\_EXP\_PER collection.

Get collection at level 2 (SDK\_BUS\_EXP\_DTL).

```

HPSAPI_SDK_BUS_EXP_SDK_BUS_EXP_PER_SDK_BUS_EXP_DTLCOLLECTION

```

```

        hSdkBusExpSdkBusExpPerSdkBusExpDtlCollection;
HPSAPI_SDK_BUS_EXP_SDK_BUS_EXP_PER_SDK_BUS_EXP_DTL
        hSdkBusExpSdkBusExpPerSdkBusExpDtl;
hSdkBusExpSdkBusExpPerSdkBusExpDtlCollection =
SdkBusExpSdkBusExpPer_GetSdkBusExpDtl(hSdkBusExpSdkBusExpPer);

```

Get and print properties at level 2.

```

for (int i211 = 0; i211 <
    SdkBusExpSdkBusExpPerSdkBusExpDtlCollection_GetCount
    (hSdkBusExpSdkBusExpPerSdkBusExpDtlCollection); i211++)
{
    hSdkBusExpSdkBusExpPerSdkBusExpDtl =
        SdkBusExpSdkBusExpPerSdkBusExpDtlCollection_Item
        (hSdkBusExpSdkBusExpPerSdkBusExpDtlCollection, i211);

    wprintf(L"oSdkBusExpSdkBusExpPerSdkBusExpDtl.SdkChargedDt:
        %s\n", printProperty
        (SdkBusExpSdkBusExpPerSdkBusExpDtl_GetSdkChargedDt
        (hSdkBusExpSdkBusExpPerSdkBusExpDtl), tmpValue));
}

```

Similar code is generated for the properties SDK\_EMPID, SDK\_EXP\_PER\_DT, SDK\_EXPENSE\_CD, SDK\_EXPENSE\_AMT, SDK\_CURRENCY\_CD, SDK\_BUS\_PURPOSE, and SDK\_DEPTID.

```

    }
}

```

Disconnect from the PeopleSoft application server by using the disconnect method of the Session object. This method clears the buffers and releases the memory.

```

//***** Disconnect from the App Server *****
Session_Disconnect(hSession);

return;
}

```



## CHAPTER 6

# Programming Component Interfaces in COM

This chapter discusses how to:

- Build the component interface APIs.
- Set up the COM environment.
- Generate a Visual Basic runtime code template.
- Use and understand the generated Visual Basic code.

---

## Building APIs for COM

If you plan to access your component interface from a COM external application, you must create a component interface API. The generated APIs are in the form of registry entries and type library files.

To build the component interface bindings:

1. Open any component interface definition in PeopleSoft Application Designer.  
Use any component interface definition, because you can build APIs for all of them, regardless of which one is open.
2. Select Build, PeopleSoft APIs.  
The Build PeopleSoft API Bindings dialog box appears.
3. Select the Build check box in the COM Type Library group box.
  - a. For the target directory, enter the directory in which you want the COM type library to be created, typically `<PS_HOME>\bin\client\winX86`.
  - b. Enter the COM server DLL location to specify where the PeopleSoft API Adapter (`psapiadapter.dll`) is typically located: `<PS_HOME>\bin\client\winX86`.
4. (Optional) Select the AutoRegister check box to execute the registry file immediately upon building the API.  
This causes your client machine registry to be immediately updated without having to register the registry entry manually.
5. (Optional) Select the Clean-up Registry check box to clean up the registry if you've previously generated the `Peoplesoft_Peoplesoft.reg` file.  
This is needed so that the older registry settings don't remain and conflict with settings made by the latest version.
6. Click OK to build the bindings that you selected.

The files that constitute the bindings are built in the location that you specified. If the operation was successful, a Done message appears in the PeopleSoft Application Designer Build window and the client machine should contain a Peoplesoft\_Peoplesoft.reg and PeopleSoft\_PeopleSoft.tlb file.

---

## Setting Up the COM Environment

When deploying component interfaces on a local client machine or web server with COM bindings, you need to have the following:

- The third-party COM application.
- The PeopleSoft application server and database.
- The Java Virtual Machine (JVM) supplied with Sun Microsystems' Java Development Kit (JDK), found in the %PS\_HOME%\JRE directory.
- A copy of the type library called PeopleSoft\_PeopleSoft.tlb that you generated during the Build PeopleSoft API Bindings process.

This type library is not specific to a single database instance—it is specific to those database objects.

- A copy of the registry file called PeopleSoft\_PeopleSoft.reg that you generated during the Build PeopleSoft API Bindings process.

This registry file is not specific to a single database instance.

To set up your client machine to access the component interface API, using COM:

1. Install the PeopleSoft File Server.

See *PeopleSoft 8.4x Installation Guide*, “Using the PeopleSoft Installer.”

2. Set the environment variable PATH to include the directory containing jvm.dll.

For example, c:\bea\jdk131\jre\bin\classic; or, if the PeopleTools installation is done locally, the path is <PS\_HOME>\jre\bin\hotspot.

3. Set the environment variable CLASSPATH to include the file psjoa.jar (typically <PS\_HOME>\class\psjoa.jar).

---

**Note.** The following steps assume you're using Microsoft Visual Basic. If you use a different compiler, apply the equivalent settings for that product.

---

4. Open the Visual Project File pscitester.vbp or sdk\_bus\_exp.vbp in Microsoft Visual Basic.
5. Select Project, References, and add the Peoplesoft\_PeopleSoft.tlb type library.

### Third-Party Application

Copy the type library and registry files to the directory containing the external API on each client machine from which you want to use the COM API.

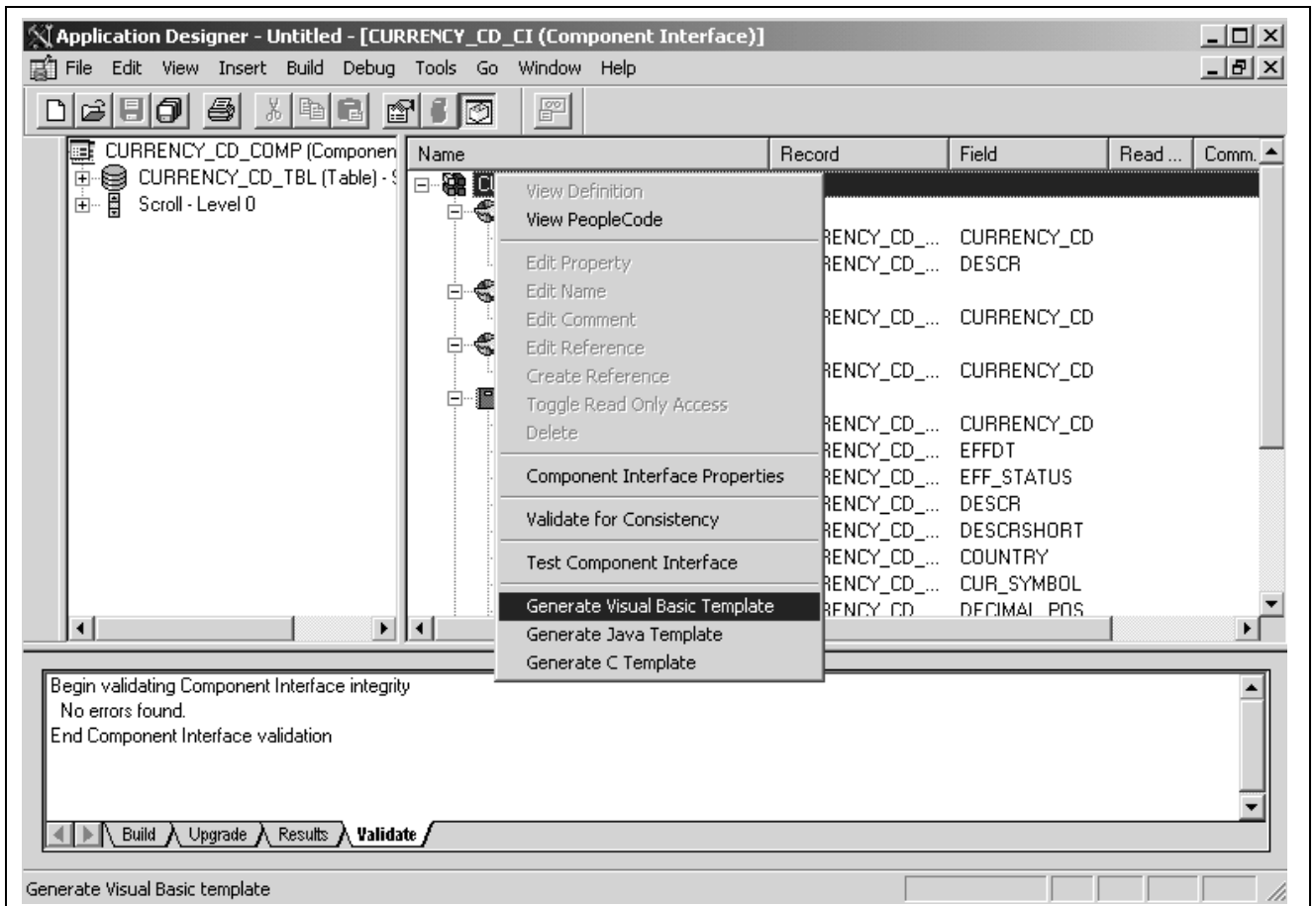
Apply the API registry settings by double-clicking Peoplesoft\_Peoplesoft.reg.

**Warning!** The registry file includes references to the External API and type library files and their locations on the original client machine where they were built, so those files must be in the same locations on the current client. If the directory structure is different, you must edit Peoplesoft\_Peoplesoft.reg to reflect the current machine before you apply the registry settings.

- If your program is early-binding, the code contains a direct reference to the path of the type library. Therefore, as you deploy, you must have the type library in the same directory on each machine.
- If your program is late-binding, the code does not contain a reference to the path of the type library. The code looks in the registry for the path to the type library. Therefore, as you deploy, you can have the type library in different directories on each machine. You must update the registry settings as part of the deployment. This is a more flexible approach.

## Generating a Visual Basic Runtime Code Template

When you want to access a component interface through PeopleSoft APIs by using Microsoft Visual Basic, PeopleSoft Application Designer generates a template in the form of boilerplate Visual Basic code that you can adapt to your purposes. This section describes how to generate the template code.



Generating Visual Basic template

To generate a Visual Basic template for your component interface:

1. Open the desired component interface definition in PeopleSoft Application Designer.
2. Right-click anywhere in the definition view to display the pop-up menu.
3. Select Generate Visual Basic Template.

When the template is successfully generated, a message appears stating the name and location of the template file.

---

**Note.** The template file is generated in the directory specified by the TEMP or TMP system environment variable on your client machine.

---

4. Open the generated file and modify the source code as needed.

---

## Understanding the Visual Basic Template

You can use the Visual Basic template as a starting point for your Visual Basic program. This section contains a skeleton of the generated Visual Basic template for a component interface named SDK\_BUS\_EXP, which is part of the component interface SDK. The template has been edited for length.

Declare the Session object.

```
Dim oSession As PeopleSoft_PeopleSoft.Session
.....

Private Sub main()
    On Error GoTo ErrorHandler
    '***** Set Connect Parameters *****
    Dim strServerName As String, strServerPort As String,
        strAppServerPath As String
    Dim strUserID As String, strPassword As String
    .....

    'Build Application Server Path
    strAppServerPath = strServerName & ":" & strServerPort
```

---

**Note.** To enable Jolt failover and load balancing in the PeopleSoft Internet Architecture, you can supply multiple application server domains for the strAppServerPath variable. Separate the domain names with a comma, and make sure there are no spaces. For example:

```
strAppServerPath = //APPSRVR1:8000, //APPSRVR2:9000
```

---

Create the PeopleSoft Session object to enable access to the PeopleSoft system. The Session object controls the environment and allows you to do error handling for all APIs from a central location.

```
'***** Create PeopleSoft Session Object *****
Set oSession = CreateObject("PeopleSoft.Session")
```

Connect to the PeopleSoft application server by using the Connect method.

```
'***** Connect to the App Server *****
If Not oSession.Connect(1, strAppServerPath, strUserID, strPassword, 0) Then
```

```

        Err.Raise 1001, "", "Unable to connect to Application Server"
        Call ErrorHandler()
        Exit Sub
    End If

```

Get a reference to the component interface providing its name. A runtime error occurs if the component interface does not exist.

```

Dim oSdkBusExp As SDK_BUS_EXP
Dim ciName As String
ciName = "SDK_BUS_EXP"
Set oSdkBusExp = oSession.GetCompIntfc(ciName)
If oSdkBusExp Is Nothing Then
    Err.Raise 1001, "", "Unable to Get Component Interface " & ciName
    Call ErrorHandler()
    Exit Sub
End If

'***** Set the Component Interface Mode *****
oSdkBusExp.InteractiveMode = False
oSdkBusExp.GetHistoryItems = True
oSdkBusExp.EditHistoryItems = False

```

Set the keys for the component interface. In this example, SDK\_EMPLID is the Get key.

```

'***** Set Component Interface Get/Create Keys *****
Dim strSDK_EMPLID As String
strSDK_EMPLID = InputBox("Enter SDK_EMPLID: ")
oSdkBusExp.SDK_EMPLID = strSDK_EMPLID

```

The Get method retrieves data from the database associated with the key values.

```

'***** Execute Get *****
If Not oSdkBusExp.Get() Then
    Err.Raise 1001, "", "No rows exist for the specified keys. Failed to get the⇒
Component Interface"
Call ErrorHandler()
    Exit Sub
End If

```

Get and print properties at level 0.

```

Debug.Print "oSdkBusExp.SDK_NAME: " & oSdkBusExp.SDK_NAME
oSdkBusExp.SDK_NAME = <*>

```

Similar code is generated for the properties SDK\_BIRTHDATE and SDK\_DEPTID.

Get the collection at level 1 (SDK\_BUS\_EXP\_PER).

```

Dim oSdkBusExpPerCollection As
    SDK_BUS_EXP_SDK_BUS_EXP_PERCollection
Dim oSdkBusExpPer As SDK_BUS_EXP_SDK_BUS_EXP_PER
Set oSdkBusExpPerCollection = oSdkBusExp.SDK_BUS_EXP_PER

```

Get and print properties at level 1.

```

Dim i17 As Integer
For i17 = 1 To oSdkBusExpPerCollection.Count
    Set oSdkBusExpPer = oSdkBusExpPerCollection.Item(i17)

```

```

Debug.Print "oSdkBusExpPer.SDK_EXP_PER_DT: " &
oSdkBusExpPer.SDK_EXP_PER_DT

```

Similar code is generated for the properties SDK\_EMPLID and SDK\_BUS\_EXP\_SUM in the SDK\_BUS\_EXP\_PER collection.

Get collection at level 2 (SDK\_BUS\_EXP\_DTL).

```

Dim oSdkBusExpDtlCollection As
SDK_BUS_EXP_SDK_BUS_EXP_PER_SDK_BUS_EXP_DTLCollection
Dim oSdkBusExpDtl As
SDK_BUS_EXP_SDK_BUS_EXP_PER_SDK_BUS_EXP_DTL
Set oSdkBusExpDtlCollection = oSdkBusExpPer.SDK_BUS_EXP_DTL

```

Get and print properties at level 2.

```

Dim i211 As Integer
For i211 = 1 to oSdkBusExpDtlCollection.Count
Set oSdkBusExpDtl = oSdkBusExpDtlCollection.Item(i211)
Debug.Print "oSdkBusExpDtl.SDK_CHARGE_DT: " &
oSdkBusExpDtl.SDK_CHARGE_DT

```

Similar code is generated for the properties SDK\_EMPID, SDK\_EXP\_PER\_DT, SDK\_EXPENSE\_CD, SDK\_EXPENSE\_AMT, SDK\_CURRENCY\_CD, SDK\_BUS\_PURPOSE, and SDK\_DEPTID.

```

Next
Next

```

Disconnect from the PeopleSoft application server by using the disconnect method of the Session object. This method clears the buffers and releases the memory.

```

'***** Disconnect from the App Server *****
If Not oSession Is Nothing Then
oSdkBusExpPer.Disconnect
Set oSession = Nothing
End If
Exit Sub
.....
End Sub

```

## CHAPTER 7

# Using the Component Interface Software Development Kit

This chapter discusses how to:

- Set SDK prerequisites.
- Use the SDK\_BUS\_EXPENSES test page.
- Test the SDK\_BUS\_EXP component interface.
- Use the component interface SDK Java sample.

---

## Understanding the Component Interface SDK

The PeopleSoft Integration SDK (Software Development Kit) is installed with the PeopleTools installation. It provides resources to assist you in developing and testing component interface-based integration between PeopleSoft and third-party applications. The SDK contains sample definitions with data and source code. For easy identification, all of the definition names start with *SDK\_*. The SDK is installed in the PeopleSoft home directory (PS\_HOME) under *sdk*.

---

**Note.** The SDK definitions and associated data are for development purposes only and should not be used in a production environment.

---

## Component Interface SDK Samples

Programming samples for the component interface SDK\_BUS\_EXP are part of the SDK. The samples are available in four languages—Java, C++, VBA, and ASP.

The component interface source code is located in the <PS\_HOME>\SDK\PSCOMPINTFC directory.

---

**Note.** The source files mentioned in this section are located relative to the installed PeopleSoft home directory (PS\_HOME).

---

---

## Setting SDK Prerequisites

To call a PeopleSoft component interface, you must have:

- A working understanding of PeopleTools components.
- A working understanding of Java, C++, or COM.
- The PeopleSoft application server and database installed.

- The Java Virtual Machine (JVM) installed that is supplied with Sun Microsystems' Java Development Kit (JDK), found in the %PS\_HOME%\JRE directory.

## Using the SDK\_BUS\_EXPENSES Test Page

The SDK includes a component interface, called SDK\_BUS\_EXP, which is part of the sample development project and is delivered with the SDK. It is built on the component SDK\_CI\_SAMPLES, which contains the page SDK\_BUS\_EXPENSES. The page exposes information about employee business expenses for external access.

**Note.** The component SDK\_CI\_SAMPLES is a sample and is not for business use.

**SDK Business Expenses**

**THIS PAGE IS PART OF THE SDK AND IS NOT FOR BUSINESS USE**

Peppen, Jacques      Employee ID: 6602      Date of Birth: 05/01/1950      DeptID: 21700

Business Expense Periods      Find | View All      First 1 of 1 Last

\*Expense Period End Date:       + -

Expense Period Total:      USD      Insert Row

Business Expense Details      Find | View All      First 1 of 1 Last

*Charge Dt	Expense Code	Expense Amount	USD	Business Purpose	Department
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Save      Return to Search      Notify

SDK\_BUS\_EXPENSES page

To test the SDK\_BUS\_EXPENSES test page:

1. Provide access to the SDK\_CI\_SAMPLES component, using PeopleTools security.
2. Select PeopleTools SDK, PeopleTools SDK, Use, SDK CI Samples.
3. Search for and select an employee ID.

## Testing the SDK\_BUS\_EXP

To test the SDK\_BUS\_EXP component interface:

1. View the component interface definition through the PeopleSoft Application Designer.

2. Test the component interface definition, using the component interface tester.

---

## Using the SDK Java Sample

The component interface sample program for Java is provided as part of the component interface SDK, located in `<PS_HOME>\sdk\pscompintfc\src\java\samples\sdk_bus_exp`.

The Java source code for the sample is in the following file:

```
sdk_bus_exp.java
```

Before you run the sample, you must build the APIs and set up the Java environment.

## Running the Java Sample

To run the compiled Java component interface sample:

1. In a DOS window, change directories to the location of the Java `sdk_bus_exp` directory  
`cd %PS_HOME%\sdk\pscompintfc\src\java\samples`
2. Launch the executable with `javasdk_bus_exp.sdk_bus_exp`.  
 You are prompted for parameters one at a time.
3. At each prompt, enter the appropriate value and press Enter.  
 Select option 1 to sign in. You are then prompted to provide the connect information.  
 If the connect succeeds, a menu appears where you can perform Get or Find functions.
4. Get details for an employee.  
 Select option 1 to get details for an employee. You are then prompted with the different modes and the employee ID for which you want to display information. Enter the employee ID 8001 and press Enter. This presents you with the level 0 data and the options that you can perform.
5. Select a business expense period.  
 Select option 8 to select a business expense period. Selecting this option displays a list of available business expense periods for the selected employee.  
 Select the expense period that you want to work with.
6. Select a business expense detail.  
 Select option 18 to select a business expense detail within the selected business expense period. Selecting this option displays a list of available business expense details within the selected business expense periods.

## Understanding the Java Sample Code

The following listings of code are taken from the Java sample program, `sdk_bus_exp.java`. (The code has been edited for length.)

Import all the required classes:

```
package sdk_bus_exp;
import java.io.*;
import java.util.*;
```

```
import psft.pt8.joa.*;
import PeopleSoft.Generated.CompIntfc.*;
public class sdk_bus_exp {
    . . . . .
```

Declare all the required objects. Only one active period and one active detail record are possible at any time. Users are prompted to select the needed values if they are not active. `oSdkBusExpCollection` is the collection object for the root, `SDK_BUS_EXP`, and `oSdkBusExp` is an item object for that collection:

`oSdkBusExpPerCollection` is the collection object for level 1, `SDK_BUS_EXP_PER`, and `oSdkBusExpPer` is an item object for that collection:

`oSdkBusExpDtlCollection` is the collection object for level 2, `SDK_BUS_EXP_PER_DTL`. `oSdkBusExpDtl` is an item object for that collection.

The `CompIntfPropInfoCollection` object is used to access the structure of a component interface. It is not specific to a component interface.

Declare the PeopleSoft session object.

The function `executeMethod` is used to launch the appropriate method depending upon the user input (`nMethodIn`). Options 1 to 5 are executed on the component interface itself. Options 6 to 15 are executed on the `SDK_BUS_EXP_PER` collection, and options 16 to 24 are executed on the `SDK_BUS_EXP_DTL` collection.

Option 1. Save any changes made to the component interface so far in this program.

Option 2. Cancel any changes made to the component interface so far in this program.

Option 3. The `GetPropertyByName` method is used to get the value of a property at the specified level. The `getSdkBusExpPropertyName()` function gets a list of all properties of the component interface. It lists only those properties that are relevant at this level and prompts the user to select a property. The selected property name is passed to the `GetPropertyByName` method to get the value of the property:

Option 4. The `GetPropertyInfoByName` method returns information about the selected property. The `getSdkBusExpPropertyName()` function is used to prompt the user for appropriate properties at this level. The selected property is returned into the variable `tempName`.

Option 5. `InsertBusExpDtlDefaults` is the user-defined method of the `SDK_BUS_EXP` component interface.

Option 6. Activate a business expense period by selecting option 8 before using this option. The `GetPropertyByName` method is used to get the value of a property at the specified level. The `getSdkBusExpPropertyName()` function gets a list of all properties of the component interface. It lists only those properties that are relevant at this level and prompts the user to select a property. The selected property name is passed to the `GetPropertyByName` method to get the value of the property.

Option 7. Activate a business expense period by selecting option 8. Then the `GetPropertyByName` method is used to get the value of a property at the specified level. The `getSdkBusExpPropertyName()` function gets a list of all properties of the component interface. It lists only those properties that are relevant at this level and prompts the user to select a property. The selected property name is passed to the `GetPropertyInfoByName` method, which returns an `CompoIntfcPropertyInfo` object. The function `printoutPropertyInfor()` is passed in this object to display the information, such as Format, Type, Length, and so on, for the property.

Option 8. Item for collection `SDK_BUS_EXP_PER` sets the pointer to the item number selected by the user. It also prompts the user for alternate values and then saves the new values. The `updateSdkBusExpPer()` function is used to set and save these new values.

Option 9. The function `insertSdkBusExpPer` is used to enable the user to add a new item in the `SDK_BUS_EXP_PER` collection. This function is discussed later in this section.

Option 10. The `DeleteItem` method is used to delete items from a collection. The user is prompted with a list of business expense periods by the function `selectSdkBusExpPer()`. The selected value is passed to the method `DeleteItem`:

Option 12. The `CurrentItem` method is used to get the current item for the specified collection.

Option 13. The `CurrentItemNum` method is used to number the current item for the specified collection.

Options 6 to 15 are for the `SDK_BUS_EXP_PER` collection, and options 16 to 25 are for the `SDK_BUS_EXP_DTL` collection. The logic used in the corresponding options of these collections is identical.

The `getSdkBusExpPer()` function lists business expense periods for the selected employee.

The `updateSdkBusExpPer()` function lists business expense periods for the selected employee.

The `insertSdkBusExpPer()` function lists business expense periods for the selected employee.

The functions `getSdkBusExpPer`, `updateSdkBusExpPer`, and `insertSdkBusExpPer` on the `SDK_BUS_EXP_PER` collection are identical to the functions `getSdkBusExpDtl`, `updateSdkBusExpDtl`, and `insertSdkBusExpDtl`.

This is the main method. It performs such functions as starting the session, getting the component interface, and disconnecting:

```
public static final void main(String[] args) System.out.println(" ");
System.out.println("\t l) Sign In ");
System.out.println("\t q) Quit ");
System.out.println(" ");
System.out.print("Command to execute (l, q) [1]: ");
charTemp = readCharacter();
switch (charTemp) {case 'q':case 'Q':.....
disconnectFromAppServer();
return;
default:
getConnectParameters();
if (connectToAppServer()) {
oSdkBusExp = (ISdkBusExp) oSession.getCompIntfc(m_strCIName);
while (getKeyType()) {
methodInt = selectMethod();
while (methodInt != 0) {
executeMethod(methodInt);
if (methodInt == 2) {
methodInt = 0;
} else {
methodInt = selectMethod();
.....
```

---

## Using the SDK C++ Sample

The component interface sample program for C/C++ is provided as part of the component interface SDK, located in `<PS_HOME>\sdk\pscompintfc\src\cpp\samples\sdk_bus_exp`.

The C++ source code for the sample is in the following file:

sdk\_bus\_exp\sdk\_bus\_exp.cpp

Before you run the sample, you must build the APIs and set up the C++ environment.

## Building the C++ Sample

To build the C++ component interface sample:

1. Open the sdk\_bus\_exp workspace in the Microsoft Visual C++ editor.
2. Build the project by selecting Build, Rebuild All.

## Running the SDK C++ Sample

To run the compiled C++ component interface sample:

1. In a DOS window, change directories to the location of the C++ sdk\_bus\_exp directory,
 

```
cd%PS_HOME%\sdk\pscompintfc\src\C++\samples\sdk_bus_exp\Debug
```
2. Launch the executable with sdk\_bus\_exp
 

You'll be prompted for parameters one at a time.
3. At each prompt, enter the appropriate value and press Enter:
 

Select Option 1 to sign in. You are then prompted to provide the connect information.

If the connect succeeds, a menu appears where you can perform Get or Find functions.
4. Get details for an employee.
 

Select option 1 to get details for an employee. You are then prompted with the different modes and the employee ID for which you want to display information.

Enter the employee ID 8001 and press Enter. This presents you with the level 0 data and the options that you can perform.
5. Select a business expense period.
 

Select option 8 to select a business expense period. Selecting this option displays a list of available business expense periods for the selected employee.

Select the expense period that you want to work with.
6. Select a business expense detail.
 

Select option 18 to select a business expense detail within the selected business expense period. Selecting this option displays a list of available business expense details within the selected business expense periods.

## Understanding the C++ Sample Code

The following listings of code are taken from the C++ sample program, sdk\_bus\_exp.cpp. (The code has been edited for length.)

Include all the headers

```
#ifdef PS_WIN32
#include "stdafx.h"
#endif #include "cidef.h"
#include "apiadapterdef.h"
#include "PSApiExternalLib.h"
```

```

#include "PSApiAdapterInter.h"
#include "PeopleSoft_PeopleSoft_i.h"
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <wchar.h>

```

Declare the PeopleSoft session handle.

```
HPSAPI_SESSION hSession;
```

Declare all the required objects. Only one active period and one active detail record are possible at any time. The user is prompted to select the needed values if they are not active. hSdkBusExpCollection is the collection handle for the root, and SDK\_BUS\_EXP. hSdkBusExp is an item handle for that collection.

hSdkBusExpPerCollection is the collection handle for level 1, SDK\_BUS\_EXP\_PER. hSdkBusExpPer is an item handle for that collection.

hSdkBusExpDtlCollection is the collection handle for level 2, SDK\_BUS\_EXP\_PER\_DTL. hSdkBusExpDtl is an item handle for that collection.

The function executeMethod is used to launch the appropriate method depending upon the user input (nMethodIn), and options 16 to 24 are executed on the SDK\_BUS\_EXP\_DTL collection.

Option 1. Save any changes made to the component interface so far in this program.

Option 2. Cancel any changes made to the component interface so far in this program.

Option 3. The GetPropertyByName function is used to get the value of a property at the specified level. The user selects a property from the list. The selected property name is passed to the GetPropertyByName method, which returns a handle. This handle is passed to the function PSApiGetStringValue() to get the value of the property:

Option 4. GetPropertyInfoByName returns information about the selected property. The user selects a property from the list. The selected property name is passed to GetPropertyInfoByName method to get a handle to hCompIntfcPropertyInfo. This handle is then passed to the function getPropertyInfo() to display the property information.

Option 5. InsertBusExpDtlDefaults is the user-defined method of the SDK\_BUS\_EXP component interface.

Option 6. Activate a business expense period by selecting option 8 before using this option. The user selects a property from the list provided. The selected property name is passed to the GetPropertyByName method. The method returns a handle that is passed to the function PSApiGetFloatValue for numeric properties, or to PSApiGetStringValue for string properties, to return the actual value.

Option 7. Activate a business expense period by selecting option 8 before using this option. The user selects a property from the list provided. The selected property name is passed to the GetPropertyInfoByName method, which returns a HPSAPI\_COMPINTFCPROPERTYINFO handle. This handle is passed to the function getPropertyInfo to display the information, such as Format, Type, Length, and so on, for the property.

Option 8. Item for collection SDK\_BUS\_EXP\_PER sets the pointer to the item number selected by the user. It also prompts the user for alternate values and then saves the new values. The updateBusinessExpensePeriod () function is used to set and save these new values.

Option 9. The function insertBusinessExpensePeriod is used to enable the user to add a new item in the SDK\_BUS\_EXP\_PER collection. This function is discussed later in this section.

Option 10. The DeleteItem method is used to delete items from a collection. The user is prompted with a list of business expense periods by the function selectBusinessExpensePeriod (). The selected value is passed to the method DeleteItem.

Option 12. The CurrentItem method is used to get the current item for the specified collection.

Option 13. The CurrentItemNum method is used to number of the current item for the specified collection.

Options 6 to 15 are for the SDK\_BUS\_EXP\_PER collection, and options 16 to 25 are for the SDK\_BUS\_EXP\_DTL collection. The logic used in the corresponding options of these collections is identical.

## Using the SDK COM Excel Sample

The component interface sample program for Microsoft Excel is provided as part of the component interface SDK, located in <PS\_HOME>\sdk\pscompintfc\src\com\samples\vba.

The Visual Basic source code for the sample is in the following file:

sdk\_bus\_exp.xls

Before you run the sample, you must build the APIs and set up the COM environment.

## Running the COM Excel Sample

When running the Microsoft Excel sample, you use the Get and Find sheet to find an employee.

2			
3	<b>Empl Id</b>		<b>80</b>
4			
5	<b>Last Name</b>		
6			
7	<b>Name</b>		
8	<b>Enter full Employee ID for Get function</b>		
9	<b>Enter partial or no information for Find function</b>		
10	<b>Select an Employee ID, and press Get selected after search</b>		
11	<b>Business expense Period for Schumacher,Simon emplid 8001</b>		
12		8001 ASD	Schumacher,Simon
13		8052 AVERY	Avery,Joan
14			

Sheet 2: Find and Get an Employee

To run the Microsoft Excel component interface sample:

1. Launch Microsoft Excel.
2. Open the Microsoft Excel sample spreadsheet.

The Microsoft Excel spreadsheet is located in <PS\_HOME>\sdk\pscompintfc\src\com\samples\vba\sdk\_bus\_exp. When prompted about macros, select Enable Macros.

3. Attach PeopleSoft References to the spreadsheet.

This example uses early bindings and hence requires attaching references to the spreadsheet. Select Tools, Macro, Visual Basic editor from the Microsoft Excel menu. This opens the VBA editor.

Select Tools, References from the menu. A dialog box appears, listing all the available references. Select the reference PeopleSoft\_PeopleSoft.

4. Sign in to the sdk\_bus\_exp sample.

Sheet 1 of the sdk\_bus\_exp spreadsheet is the sign-in page. Provide the connect information and press Tab to navigate out of the fields. Click Connect to establish the connection.

5. Find an employee by using the Find keys.

The Find and Get keys are located on Sheet 2.

6. Select an employee from the list.

Select an employee ID from the list by making the cell active and then pressing the GET selected button.

7. Get an Employee by providing the Get key.

Enter the complete employee ID in cell B3. Press Tab to navigate out of the cell, and click Get (EMPLID). A list of all the available periods is displayed.

8. View details.

To view the details for the listed business expense periods, click the Toggle Details button.

9. Add a new business expense period.

Click the Insert period button. This redirects you to Sheet 3. Enter the business expense period date. Press Tab and click the Save New Period button.

10. Add a new business expense detail.

Click the Insert Detail button. This redirects you to Sheet 3. Enter the charge date, expense code, amount, department ID, and business purpose. Press Tab, and save the new detail by pressing the Save New Detail button.

You can list the expense periods for the employee.

11	Business expense Period for Schumacher,Simon emplid 8001		
12	Period Number	Period Dated	Period Total
13	Period # 1	5/16/2001	1660.910034
14			
15	Period # 2	11/9/2000	690.7800293
16			

Expense periods

## Understanding the COM Excel Sample Code

The following listings of code are taken from the Microsoft Excel sample program, sdk\_bus\_exp.xls. (The code has been edited for length.)

View the code by selecting Tools, Macro, Visual Basic Editor from the menu.

Declare the PeopleSoft session object.

List Business Expense Periods, using the Item method to get a specific item of the type SDK\_BUS\_EXP\_PER.

List Business Expense Details, using the Item method to get a specific item of the type SDK\_BUS\_EXP\_DTL.

To save a new business expense period, use the InsertItem method. This method inserts a new row and returns an item of the type SDK\_BUS\_EXP\_PER. The item contains the properties. Set the properties and execute the Save method.

To save a new business expense detail, use the `InsertItem` method. This method inserts a new row and returns an item of the type `SDK_BUS_EXP_DTL`. The item contains the properties. Set the properties and execute the `Save` method.

## Using the SDK COM ASP Sample

The component interface sample program for ASP is provided as part of the component interface SDK, located in `<PS_HOME>\sdk\pscompintfc\src\com\samples\asp\sdk_bus_exp`.

The ASP source code for the samples is in these files.

Before you run the sample, you must build the APIs and set up the COM environment.

## Running the COM ASP Sample

When running the ASP sample, you use the `Get` key to find an employee.

**SDK Business Expenses Component Interface - Search**

[Sign Off](#)

**'Get' Keys for Component Interface**

SDK\_EMPLID

ASP Get key

To run the ASP component interface sample:

1. Install and configure the IIS web server.
2. Create a virtual directory to point to `<PS_HOME>\sdk\pscompintfc\src\com\samples\asp\sdk_bus_exp`.
3. Start the web server.
4. Run the SDK example through the browser.

The web address `http://machinename/sdkSDK_BUS_EXP_Signon.asp` launches the SDK application.

5. Provide the connect information and click `Submit`.
6. Get details for an employee.

Enter the `Get` key (`SDK_EMPLID`) and click the `Get` button. This lists all the business periods for the selected employee ID.

7. Update a business expense period.

Click the `Update` button to update the business expense period. Update the expense period end date and click the `Save` button.

8. Insert a business expense period.

Click the `Insert` button to update the business expense period. Add the new expense period end date and click the `Save` button.

9. Delete a business expense period.

Click the Delete button to delete the business expense period. You are prompted to decide to delete the row. Click OK to confirm the delete, and click Cancel to cancel the operation.

10. Update a business expense detail.

Select a business expense period by clicking the Update button from the business expense period row.

11. Insert a business expense detail.

Click the Insert button to insert a new business expense period. Enter the values for charge date, expense code, expense amount, currency code, business purpose, and department ID. Click the Save button to save changes.

12. Delete a business expense detail.

Click the Delete button to delete the business expense detail. You are prompted to decide to delete the row. Click OK to confirm the delete, and click Cancel to cancel the operation.

You can list the expense periods for the employee:

**Listing Details for Employee ID: 8001**

**Business Period List**

Employee Number	Period Number	Period Total	Period End Date	Work with Period
8001	1	1660.91	05/16/2001	Update Insert Delete
8001	2	223.95	11/09/2000	Update Insert Delete

Expense periods

## Understanding the COM ASP Sample Code

The following listings of code are taken from the ASP sample programs. (The code has been edited for length.)

### SDK\_BUS\_EXP\_Signon.asp

This is a sign-on page to the SDK Business Expense sample.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Provide the form action. On the submit from this page SDK\_BUS\_EXP\_GetSearchParameters.asp is called.

Click the Submit button to run a JavaScript function checkRequiredFields(), which checks that all the connect information is provided.

The connection information is forwarded to the next page, using hidden fields.

### SDK\_BUS\_EXP\_GetSearchParameters.asp

This page prompts for the Find and Get Keys. You can also set the component interface modes: Interactive Mode, Get History Items, and Edit History Items.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Sub `getSearchParameters` prompts the user for the Get or Find key and the component interface modes (interactive, get history items, and edit history items).

Sub `getSearchParameters` calls the appropriate page for Get and Find, using the JavaScript function `invokeMethod()`.

### **SDK\_BUS\_EXP\_GetSearchResults.asp**

This page is called if the Find option was selected.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the Find keys.

Get the search result.

The function `getSearchResults` lists all the employees for the provided Find keys by setting the Find keys and executing the Find method.

Loop through the collection to list all the employee IDs.

### **SDK\_BUS\_EXP\_GetBusinessExpenses.asp**

This page lists all the business expense periods for the selected employee.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the Key Field.

Set the Component Interface Get Key.

Get the business expense periods by executing the function `getBusinessExpensePeriods`.

The function `getBusinessExpensePeriods` gets the business expense period, and then loops through the collection, using the `Item` method to get a specific business expense period. Each property in that item is then displayed.

The Update button is of the type `submit`. Since the form action is set to `SDK_BUS_EXP_UpdateBusinessExpensePeriod.asp`, this page is launched. The Insert and Delete buttons use the JavaScript functions `insertBusinessExpensePeriod()` and `deleteBusinessExpensePeriod()`.

The JavaScript function `insertBusinessExpensePeriod` sets the `form.action` to `SDK_BUS_EXP_InsertBusinessExpensePeriod.asp` and submits the page.

The JavaScript function `deleteBusinessExpensePeriod` sets the `form.action` to `SDK_BUS_EXP_DeleteBusinessExpenseDetail.asp` and submits the page.

### **SDK\_BUS\_EXP\_UpdateBusinessExpensePeriod.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Update business expense details for the selected employee ID and business expense period

Submitting the `updateBusinessExpensePeriod` gets the business expense period collection. It passes `PERIODNUM` to the `Item` method to get the business expense period to be modified.

This page also lists the business expense details, using the `getBusinessExpenseDetails` function.

The Save button uses the JavaScript function `saveBusinessExpensePeriod` to save changes made to the business expense period. The function `saveBusinessExpensePeriod` sets the form action to `SDK_BUS_EXP_SaveBusinessExpensePeriod.asp` and submits the form.

### **SDK\_BUS\_EXP\_SaveBusinessExpensePeriod.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Save business expense period for the selected employee ID and business expense period. Get the specific business expense period by using the `Item` method. Set the `SDK_EXP_PER_DT` property with the new value and execute the `Save` method.

### **SDK\_BUS\_EXP\_InsertBusinessExpensePeriod.asp**

This page enables the user to insert a new business expense period.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Call the `insertBusinessExpensePeriod` function to insert a new business expense period.

Get the business expense period.

The `InsertItem` method is used to insert a new item in the `SDK_BUS_EXP_PER` collection.

A field to enter the `SDK_BUS_PER_DT` is created.

The Save button calls the JavaScript function `newBusinessExpensePeriod`. This function sets the action of the form to `SDK_BUS_EXP_NewBusinessExpensePeriod.asp` and submits the form.

### **SDK\_BUS\_EXP\_NewBusinessExpensePeriod.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include `SDK_BUS_EXP_FUNCLIB.asp`. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Get the `SDK_BUS_EXP_PER_COLLECTION`. Execute the `InsertItem` method. Set the `SDK_EXP_PER_DT` property and execute the `Save` method.

### **SDK\_BUS\_EXP\_DeleteBusinessExpensePeriod.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

The Submit deleteBusinessExpensePeriod deletes a business expense period.

This Submit gets the SDK\_BUS\_EXP\_PER collection. The DeleteItem method deletes the specified row. The Save method is executed. To view the updated SDK\_BUS\_EXP\_PER collection, execute the Cancel method. Set the keys and execute the Get method. The function getBusinessExpensePeriods displays the business expense periods.

Execute the Save method.

Execute the Cancel method

Set the component interface Get Key.

Execute the Get method.

### **SDK\_BUS\_EXP\_InsertBusinessExpenseDetail.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Sub insertBusinessExpenseDetail inserts a business expense detail for the selected business expense period.

Get the SDK\_BUS\_EXP\_PER collection in Sub insertBusinessExpenseDetail.

Get the SDK\_BUS\_EXP\_PER collection, using the Item method.

Get the SDK\_BUS\_EXP\_DTL collection.

Get the SDK\_BUS\_EXP\_DTL using the Item method.

Create a form to get the properties for SDK\_BUS\_EXP\_DTL.

The Save button calls the JavaScript function saveBusinessExpenseDetail sets action of the form to SDK\_BUS\_EXP\_SaveBusinessExpenseDetail.asp and submits the form.

### **SDK\_BUS\_EXP\_SaveBusinessExpenseDetail.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Submit saveBusinessExpenseDetail saves the business expense detail.

Submit saveBusinessExpenseDetail gets the SDK\_BUS\_EXP\_PER collection.

Get the SDK\_BUS\_EXP\_PER, using the Item method.

Get the SDK\_BUS\_EXP\_DTL collection.

Get the SDK\_BUS\_EXP\_DTL, using the InsertItem method. Set the properties.

Execute the Save method.

### **SDK\_BUS\_EXP\_DeleteBusinessExpenseDetail.asp**

This page enables the user to update the expense period data as well as insert and delete business expense details.

Include SDK\_BUS\_EXP\_FUNCLIB.asp. This file contains all the common functions.

Get the connection information forwarded from the previous page.

Get the key fields.

Sub deleteBusinessExpenseDetail deletes a business expense detail from the selected business expense period.

Get the SDK\_BUS\_EXP\_PER collection in Sub insertBusinessExpenseDetail.

Get the SDK\_BUS\_EXP\_PER collection, using the Item method.

Get the SDK\_BUS\_EXP\_DTL collection.

Execute the DeleteItem method.

Execute the Save method.



## CHAPTER 8

# Programming Component Interfaces in PeopleCode

This chapter discusses:

- PeopleCode behavior and limitations.
- PeopleCode runtime code template.
- PeopleSoft runtime code template.

---

## Understanding PeopleCode Behavior and Limitations

Note the behavior and limitations discussed in this section when you write PeopleCode for a component interface.

### PeopleCode Event and Function Behavior

PeopleCode events and functions that relate exclusively to GUI and online processing cannot be used by component interfaces. These include:

- Search dialog processing.

When you run a component interface, the SearchInit, SearchSave, and RowSelect events don't fire. This means that any PeopleCode associated with these events will not run. The first event to run is RowInit.

- Menu PeopleCode and pop-up menus.

The ItemSelected and PrePopup PeopleCode events are not supported. In addition, the CheckMenuItem, DisableMenuItem, EnableMenuItem, HideMenuItem, and UncheckMenuItem functions aren't available.

- Transfers between components, including modal transfers.

The TransferPage, DoModalPageGroup, and IsModalPageGroup functions cannot be used.

- Dynamic tree controls.

Functions related to this control, such as GetSelectedTreeNode, GetTreeNodeParent, GetTreeRecordName, RefreshTree, and TreeDetailInNode cannot be used.

- ActiveX controls.

The PSControlInit and PSLostFocus events are not supported, and the GetControl function cannot be used.

- DoSave() and DoSaveNow().

The DoSave() and DoSaveNow() pcode functions are not supported. You should use the component interface Save() method and wrap the DoSave() and DoSaveNow() functions so they don't execute when called from a component interface.

- Functions that are ignored in a component interface call.

Some PeopleCode functions are ignored if they are called through a component interface. These functions are:

- WinMessage
- CheckMenuItem
- DisableMenuItem
- EnableMenuItem
- HideMenuItem
- UncheckMenuItem
- SetCursorPos
- TransferPanel
- TransferPage
- DoModalComponent
- IsModalComponent
- DoModalPanelGroup
- IsModalPanelGroup
- GetSelectedTreeNode
- GetTreeNodeParent
- RefreshTree
- TreeDetailInNode
- GetControl
- DoSave
- DoSaveNow
- Gray
- Ungray

## CopyRowset Language Considerations

In previous PeopleSoft releases, CopyRowset\* functions for component interfaces were not sensitive to the language code on PSCAMA. Because of this, related language processing did not take place when language code on PSCAMA was different than the base language code. PeopleSoft now detects the language code in PSCAMA.

## Limitations of Client-Only PeopleCode

Component interfaces can run on either the client or the server. By default, a component interface runs on the server. It runs on the client only if the code calling the component interface is running on a client machine.

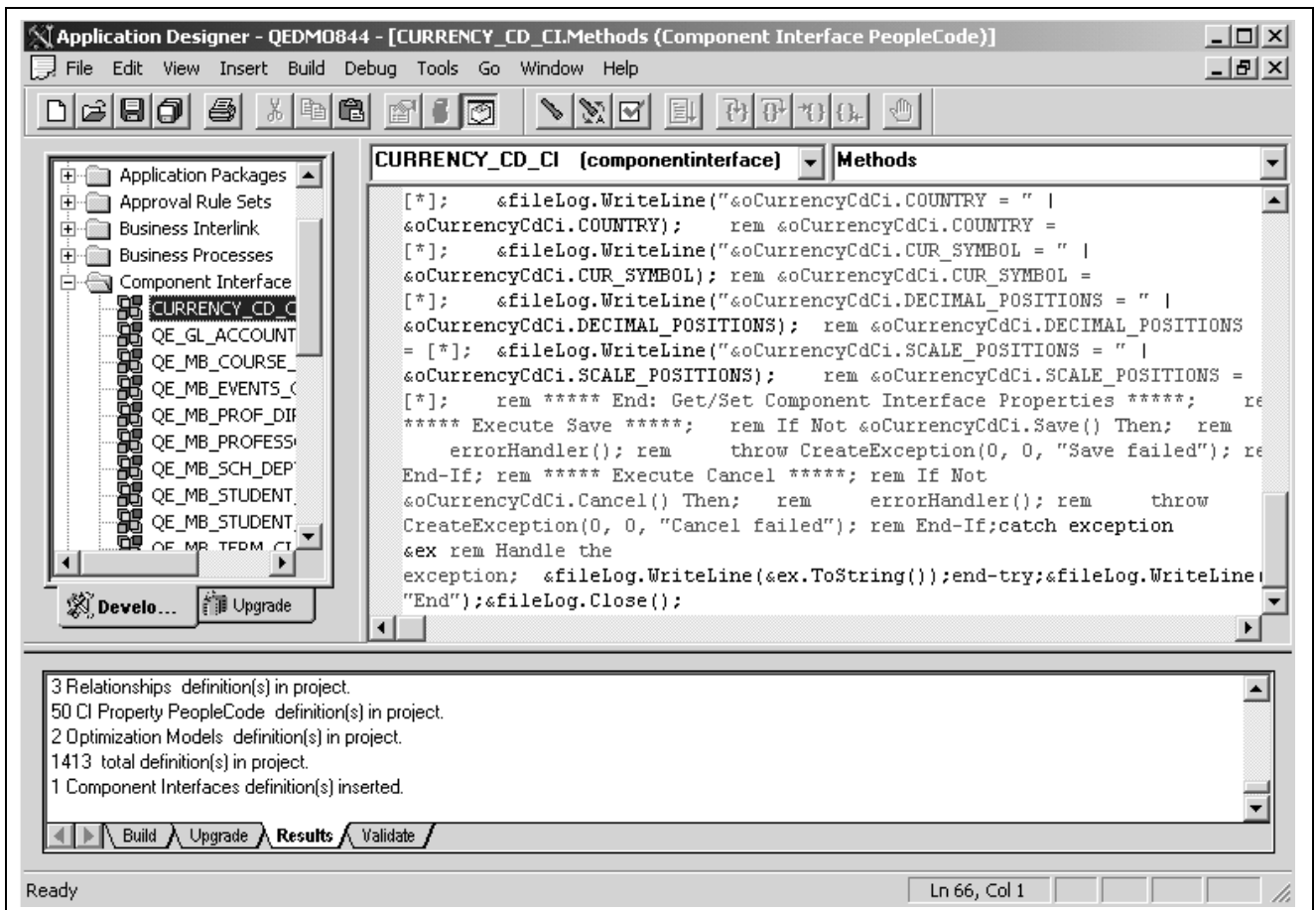
Component interfaces must run either entirely on the server or entirely on the client. To ensure this runtime restriction, component interface references declared in PeopleCode must be declared as local, not global, variables.

Some built-in functions are always client-only; others are client-only under specific conditions.

Some built-in functions behave differently when used in three-tier mode, as opposed to two-tier mode.

## Generating a PeopleCode Template

To access a component interface using PeopleCode, PeopleSoft Application Designer generates a template in the form of boilerplate PeopleCode that you can adapt to your purposes. This section describes how to generate the template code.



PeopleCode generated by dragging and dropping a component interface

To generate a PeopleCode template for a component interface:

1. Open the desired component interface definition in PeopleSoft Application Designer.
2. Insert the component interface into a project.
  - a. Select Insert, Current Object into Project.
  - b. Save the project.
3. Open the PeopleCode editor.

You can associate component interface PeopleCode with a record, a component, an application message, or an Application Engine program.
4. Select the component interface from the project workspace.

Drag and drop the object from the project into the PeopleCode editor.

5. Make any necessary changes to the PeopleCode in the PeopleCode editor window.

---

## Understanding the PeopleCode Template

The code shown in this section is a dynamically generated PeopleCode template that you can use as a starting point. Replace all default values or <\*> notations with specific values or references to valid PeopleCode variables (replace this entire three-character string: <\*> ).

---

**Note.** The requirement to populate a non-create key is no longer a requirement to do the initial save.

---

PeopleCode runs only if you are connected. This means that you do not have to explicitly connect. Instead, connect to the existing session, using the %Session system variable.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference*, “Quick Reference for PeopleCode Classes,” Session Classes Methods and Properties.

You cannot connect to a different database through PeopleCode.

Set the PeopleSoft session error message mode. This property is used to determine how messages are output. This property takes either a numeric value or a constant. The default value is 1 (%PSMessages\_CollectionOnly).

This property sets the value for the session. You can change modes during a session, for example, if you’re starting a component interface. However, after you run the component interface, you should set the value back. Here is the list of modes that you can use:

Mode Value	Purpose
0	Return no messages
1	Default. Log messages into the PSMessage collection.
2	Pop up a message dialog box.
3	Log messages into the PSMessage collection and pop up a message dialog box.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference*, “Session Class,” PSMessagesMode.

### PeopleCode Template Notes

Get a reference to the component interface providing its name. (A runtime error occurs if the component interface does not exist.)

Set the keys for the component interface. In this example SDK\_EMPLID is the Get key.

The get() method retrieves data from the database, associated with the key values.

Get and print properties at level 0.

Similar code is generated for the properties SDK\_BIRTHDATE and SDK\_DEPTID.

Get collection at level 1 (SDK\_BUS\_EXP\_PER).

Get and print properties at level 1.

Similar code is generated for the properties SDK\_EMPLID and SDK\_BUS\_EXP\_SUM in the SDK\_BUS\_EXP\_PER collection.

Get collection at level 2 (SDK\_BUS\_EXP\_DTL).

```
&oSdkBusExpDtlCollection = &oSdkBusExpPer.SDK_BUS_EXP_DTL;
```

Get and print properties at level 2.

Similar code is generated for the properties SDK\_EMPID, SDK\_EXP\_PER\_DT, SDK\_EXPENSE\_CD, SDK\_EXPENSE\_AMT, SDK\_CURRENCY\_CD, SDK\_BUS\_PURPOSE, and SDK\_DEPTID.



## CHAPTER 9

# Using the Excel to Component Interface Utility

This chapter discusses how to:

- Set up connection information.
- Create a template.
- Manage templates.
- Stage and submit data.
- Resubmit corrected data.
- Construct the SOAP/XML request.
- Send the SOAP/XML request and receive the SOAP/XML response.

---

## Understanding the Excel to Component Interface Utility

Use the Excel to Component Interface utility to upload data from Microsoft Excel into your PeopleSoft database, using component interfaces. Each source workbook contains both worksheets and Excel Visual Basic code modules that execute business logic for each transaction.

Use the Microsoft Excel workbooks as a template to create worksheets that are specific to the business logic that you need to use when you are uploading data to your PeopleSoft system. You can copy the Data Input sheet to other workbooks for distribution without copying the code modules.

The code formats spreadsheet data into a PeopleSoft readable Document Object Model (DOM) structure, and submits it to the PeopleSoft database. Next a PeopleCode program parses the DOM structure and uses the component interface to create entries in the PeopleSoft database, validating the data submitted against the business logic built into the PeopleSoft component. Because the component interface is a wrapper around the component, all logic applied during data entry is applied when loading data through this tool.

The component interface executes all the necessary PeopleCode events and the field-level edits. Based upon results from saving the component interface, another DOM is created in the PeopleCode that returns success, warnings, and/or errors to the Microsoft Excel document. Records in error can be corrected and resubmitted.

### Microsoft Excel Column and Row Restrictions

A Microsoft Excel spreadsheet has a physical limitation of 252 columns and 65,000 rows. When you are creating a template and submitting data, you will need to keep these restrictions in mind.

When a template has more than 252 properties, you can work around this limitation by deselecting an appropriate number of columns to meet the restriction before submitting to the database. You can determine in advance which fields should be selected as input cells, which cells should be included for submission, and which cells need not be included at all.

To work around Microsoft Excel's row limitations, you may need to stage your data in batches so that the number of rows on the Staging and Submission sheet will not exceed 65,000.

---

**Note.** You will find the optimal performance when using this utility with small to medium-complexity component interfaces. For large component interfaces, other methods of uploading may be more appropriate, including File Layout and Application Engine.

In addition, because of Microsoft Excel's high memory requirements, performance with more complex component interfaces will be inhibited. The number of cells containing values on any one sheet should be less than 1,000,000. Failure to ensure this can potentially result in serious performance issues with Microsoft Excel, including crashes.

---

## Software Requirements

This utility is available only for Microsoft Office 2000 and specifically needs the Microsoft XML parser, msxml4.dll. Some applications have been known to overwrite previous versions of this file with another version, occasionally causing the utility not to function properly. You should use MSXML version 4, Service Pack 2, because it can coexist with older versions of the parser. You can download and reinstall MSXML from the Microsoft website.

The Excel to Component Interface utility requires the Visual Basic 6.0 SP5: Run-Time Redistribution Pack and MSXML (Microsoft XML Parser) 4.0 Service Pack 4.

See the Microsoft downloads website for more information on these service packs.

## Building a Component Interface for the Excel to Component Interface Utility

To use the Excel to Component Interface utility effectively, you must have a complete understanding of the component you are using and the component interface that is built around it. When you build a component interface, you need to understand what data needs to be entered and which fields on the component need to be exposed as component interface properties. Fields that are not relevant for data input should not be exposed on the component interface. This reduces processing time when loading data, as well as saving time when building the template because there will be no need to delete unnecessary properties on the template.

Some component interface structures will need to be modified before they can be used to load data through the utility. Components that have logic to insert multiple rows in child collections, and then require more values to be set on those collections, will need modification to the component to work with the Excel to Component Interface utility. Change the component so that the logic to insert and partially populate these rows does not happen by default through the component interface.

`%CompIntfc` and `%CompIntfcName` can be used so that this logic does not fire either from any component interface or from the component interface that you created for use with the Excel to Component Interface utility.

Additionally, components that have no keys at level 0, but rely on logic at level 0 to load the level 1 collection, cannot be loaded by using the Excel to Component Interface utility.

Component interfaces that rely on `CommitWork` to save the data cannot be used in the Excel to Component Interface utility.

Prompt and translate table values are validated when data is saved and submitted to the database through the Excel to Component Interface utility. This is different from the behavior on the page when prompts and translates are validated interactively. Some components may use prompts that are dynamically populated. For those situations, you must know what the valid values for the prompt will be.

---

**Note.** Remember that any changes made to the structure of a component interface will need to be reflected in the template as well. Always ensure that the component interface and the template in the Excel to Component Interface utility are in sync. Structural changes made in only the component interface will cause an error in the Excel to Component Interface utility when data is submitted to the database.

---

## Testing the Component Interface

Before using the Excel to Component Interface utility, you should run the component interface through the component interface tester in three-tier mode. Testing the component interface using the tester enables you to troubleshoot any problems before running the component interface through the utility. If the component interface does not work in the tester, it will not work in the Excel to Component Interface utility, either. The component interface tester is located on the Tools menu in PeopleSoft Application Designer.

See [Chapter 3, “Developing the Component Interface,” Testing the Component Interface, page 44.](#)

## Performance Expectations

The performance of a component interface depends entirely upon the underlying component. If the component has a complex user interface with many pages and scrolls, the component interface generally will have a slower processing time. The best performance times are found with small and medium-complexity component interfaces.

## PeopleCode Behavior and Limitations

Certain PeopleCode functions and events that are specific to the user interface do not execute through the component interface. You will need to modify PeopleCode for the component, pages, and records when you build the component interface for the component.

PeopleCode events and functions that relate exclusively to the page interface and online processing cannot be used by component interfaces. These include:

- Search dialog processing.
- Menu PeopleCode and pop-up menus.
- Transfers between components, including modal transfers.
- Dynamic tree controls.
- ActiveX controls.
- DoSave and DoSaveNow.
- Functions that are ignored in a component interface call.

See [Chapter 8, “Programming Component Interfaces in PeopleCode,” Understanding PeopleCode Behavior and Limitations, page 91](#) and *Enterprise PeopleTools 8.45 PeopleBook: PeopleCode API Reference*, “Component Interface Classes,” [Understanding Component Interface Class](#).

## Automatic Defaulting of Properties

When you create a new component interface in PeopleSoft Application Designer, the system can create default properties for all the fields exposed on the component interface that meet certain criteria.

When creating a new component interface, the following requirements must be met to qualify as a default property.

The fields should be of the following types:

- Character
- Long character
- Number
- Signed number
- Date
- Time
- Datetime

The field should be one of the following page control types and must be exposed on the page:

- Edit box
- Drop-down list box
- Check box
- Radio button

See [Chapter 3, “Developing the Component Interface,” Creating a New Component Interface, page 9](#).

## Getting Started with the Excel to Component Interface Utility

To launch the Excel to Component Interface utility, you will need to locate the ExcelToCI.xls executable file in the PS\_HOME\excel folder.

In addition, you will need to perform the following tasks:

- Grant access to the iScript WEBLIB\_SOAPTOCI.
- Enable macros in Microsoft Excel.

### Granting Access to WEBLIB\_SOAPTOCI

To use the Excel to Component Interface utility, you must grant access to the iScript WEBLIB\_SOAPTOCI in the permission list of the user who is building the template.

See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Setting Up Permission Lists”.

### Enabling Macros in Excel

The Excel to Component Interface utility relies on macros; therefore, you must enable macros in Microsoft Excel for the utility to work. When a Microsoft Excel spreadsheet is opened, the system displays a dialog box asking you to select whether or not macros should be enabled on the spreadsheet. Always select Enable Macros so that the macros delivered with the Excel to Component Interface utility can function.

To ensure that the macros are available to run, you will need to set the security level in Microsoft Excel to allow macros to open.


To enable macros in Excel:

1. Open the Excel to Component Interface utility.
2. From the Excel menu, select Tools, Macros, Security.
3. Select either Medium or Low to enable the macros.
4. Select OK.

## Viewing the Coversheet

The coversheet of the Excel to Component Interface utility workbook gives a brief overview of the process flow and functionality of the tool.

Access the Coversheet tab in ExcelToCI.xls:


ExcelToCI

**The purpose of this workbook is to upload data from Excel into PeopleSoft using the Component Interface to execute business logic for each transaction. This source workbook contains both worksheets and Excel VBA code modules. The Worksheets can be copied to other workbooks for distribution without copying the code modules.**

**Process:**

- 1) **Connection Information sheet:** The information provided on this sheet is required to create a new template or submit data to the database. You will need to specify environment information as well as information in regard to how each transaction should be handled. The Action will be filled in automatically.
- 2) **Template sheet:** Here you create the template that you are going to use. The template is based upon the structure of a Component Interface on the PeopleSoft database.
  - 2.1. New Template:** When prompted, enter the name of your PeopleSoft User ID, password, and the Component Interface you wish to use. The connect information previously provided is used to retrieve the PeopleSoft Component Interface properties. The Component Interface structure is displayed graphically.
  - 2.2. Manipulate Template:** Here input cells are selected for inclusion on the Data Input and Staging & Submission sheets. The purpose of each button provided to allow manipulation of the template is more fully described by mousing over the button on the toolbar.
  - 2.3. New Data Input:** This button copies the selected input cells into the Data Input sheet. The Data Input sheet becomes active, and you will be prompted before the structures and data on that sheet are deleted.
- 3) **Data Input sheet:** Here you enter data values for submission to the PeopleSoft database.
  - 3.1. Stage Data for Submission:** The data entered on the Data Input sheet is then staged in hierarchical form in preparation for submission to the database.
- 4) **Staging & Submission sheet:** The last step is to format the data and submit to the database.
  - 4.1. Submit Data:** This submits the data to the database. Your PeopleSoft User ID and Password will be required. Each row submitted is marked with the reply from the database which will either be OK, Warning, or Error.
  - 4.2. Post Results:** This posts the status for each row submitted to the database on the Data Input sheet so that data in a status of error can be corrected and resubmitted.

Coversheet tab

---

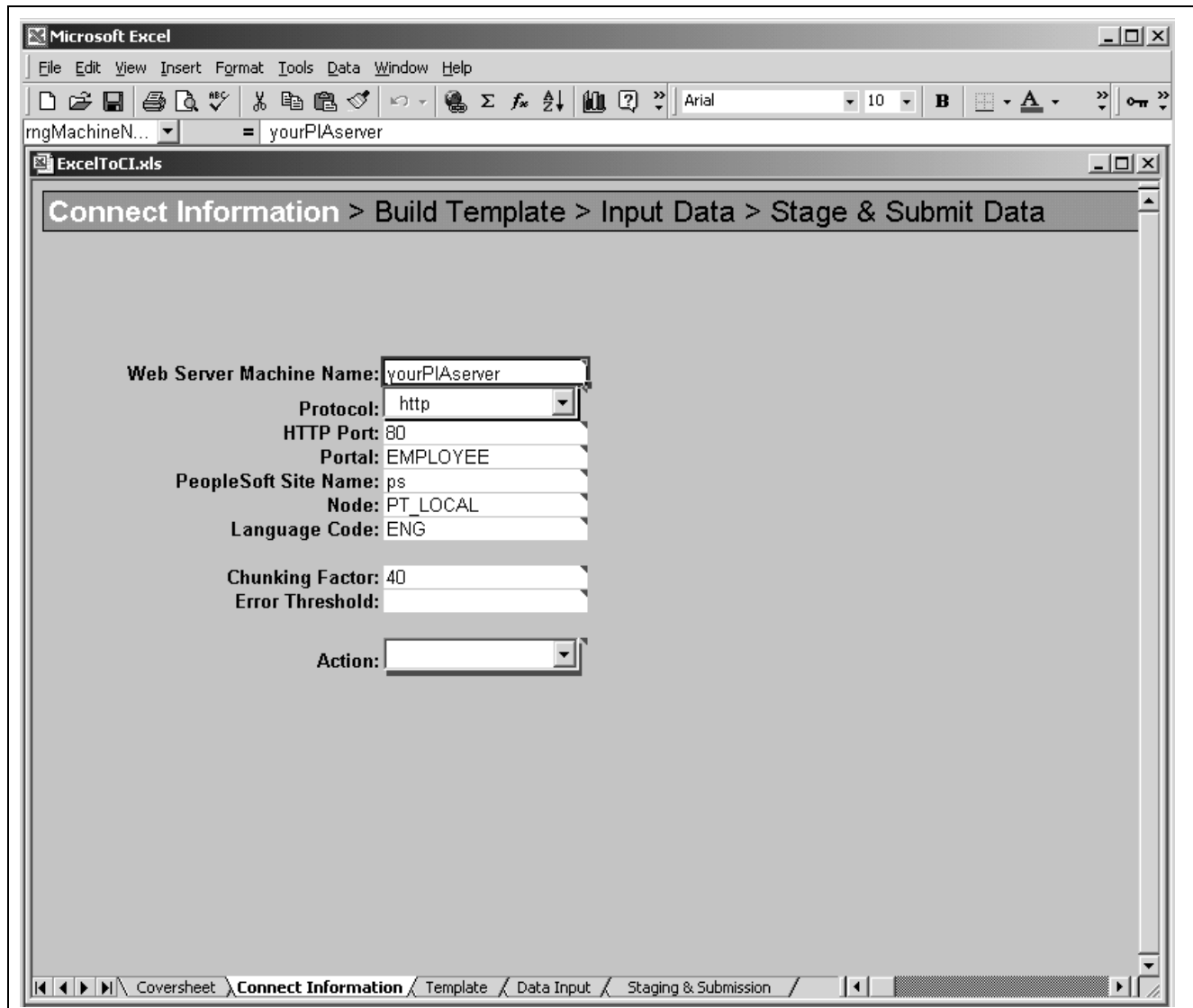
## Setting Up Connection Information

This section covers:

- Entering connection information.
- Connecting to the database.

### Entering Connection Information

Access the Connect Information tab in ExcelToCI.xls:



Connection Information tab

The information on this page is required to create a new template or to submit data to the database. You will need to specify environment information as well as information about how data should be transmitted. The Action field will be filled in automatically based on your setup and the component interface that the template is associated with.

The initial connection settings will be the PeopleSoft default values. You will need to modify these values for your specific implementation of PeopleSoft. If you are unsure what to enter for these values, check with your system administrator.

The connection options are:

**Web Server Machine Name** The name of the PeopleSoft web server that you are connecting to.

**Protocol** The protocol used to access the web server. The default is http. The preferred protocol is https.

**HTTP Port** The HTTP port number that the web server uses. The default is 80.

<b>Portal</b>	The name of the portal you are using. <i>EMPLOYEE</i> is a default portal shipped with PeopleSoft.
<b>PeopleSoft Site Name</b>	The PeopleSoft site name that you entered when you installed the PeopleSoft Internet Architecture. The default is <i>ps</i> .
<b>Node</b>	The PeopleSoft default local node name. The default is <i>PT_LOCAL</i> .
<b>Language Code</b>	The code for the language that the data is submitted to the database in and the template is created in. If no language code is specified, the base language is used.
<b>Chunking Factor</b>	The number of rows of data to be transmitted to the database at one time. The default is 40.
<b>Error Threshold</b>	The total number of errors that are permitted before submission to the database ceases. When the error threshold is exceeded, an error message appears and submission to the database stops.
<b>Action</b>	<p>The value for the Action field is populated by the system when the component interface is retrieved from the database. However, you can change the populated value by selecting it from the Action drop-down list.</p> <p>The types of actions available are based on the structure of the component interface. The different actions are:</p> <ul style="list-style-type: none"> <li>• <i>Create</i> if the component interface has create keys. Use this mode when new keys are being added at level 0.</li> <li>• <i>Update</i> if the component interface does not have create keys. Use this mode if you are adding new children to an existing parent.</li> <li>• <i>UpdateData</i> requires you to select this option from the drop-down list. This mode is used to update specific non-key values that already exist but need to be updated. The system uses the keys to locate the row, and when a match is found, the row is updated with new data. If a key match is not found by the system, it displays an error message indicating which collection was missing a key match.</li> </ul>

See *Enterprise PeopleTools 8.45 PeopleBook: Internet Technology*, “Understanding PeopleSoft Pure Internet Architecture” and *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Understanding PeopleSoft Security”.

## Multilingual Support

You can use the Excel to Component Interface utility to upload data from any installed language. For each delivered language, PeopleSoft delivers a translated version of the Excel to Component Interface worksheet. When you change the language code on the Connect Information, the field labels on the template and any error messages on the Staging and Submission sheet appear in the language that you specified. When data is submitted, the standard PeopleSoft related language architecture processing occurs. You will find the translated worksheets in the appropriate language directory found in the PS\_HOME/EXCEL directory.

If you want to submit data in several different languages, then a separate submission must be done for each language.

## Error Thresholds and Chunking

A running error count is kept for each chunk of data that is being submitted to the database. When the total error count exceeds the error threshold that you specified on the Connection Information tab, submission to the database stops and the system displays an error message. Rows that errored out will have a status of *Error* on the Data Input page and should be corrected. The data submitted to the database before the error threshold was reached will remain in the target database. Rows not yet submitted will be submitted when the data is restaged and submitted.

## Connecting to the Database to Create a Template and Submit Data

Your PeopleSoft login information is needed for both creating the template and submitting data to the database.

Access the Login dialog box by selecting the Template tab and then selecting the New Template button, or by selecting the Submit Data button on the Staging and Submission tab:

Login dialog box

The system uses your user ID and password to ensure that you have the correct permissions to access the component interface that you are creating the template on. You must be granted permission to access the component interface that you are using.

<b>User ID/Password</b>	Enter your PeopleSoft user ID and password.
<b>Component Interface Name</b>	Enter the name of the component interface name for which you want the template created.
<b>Generate Log</b>	Select the Generate Log check box to create one log file for ExcelToCI.xls and one for the SOAPTOCI Web Library.

---

**Note.** Unless you are troubleshooting errors, you should run the Excel to Component Interface utility without creating log files. Logs should be generated for debugging purposes only.

---

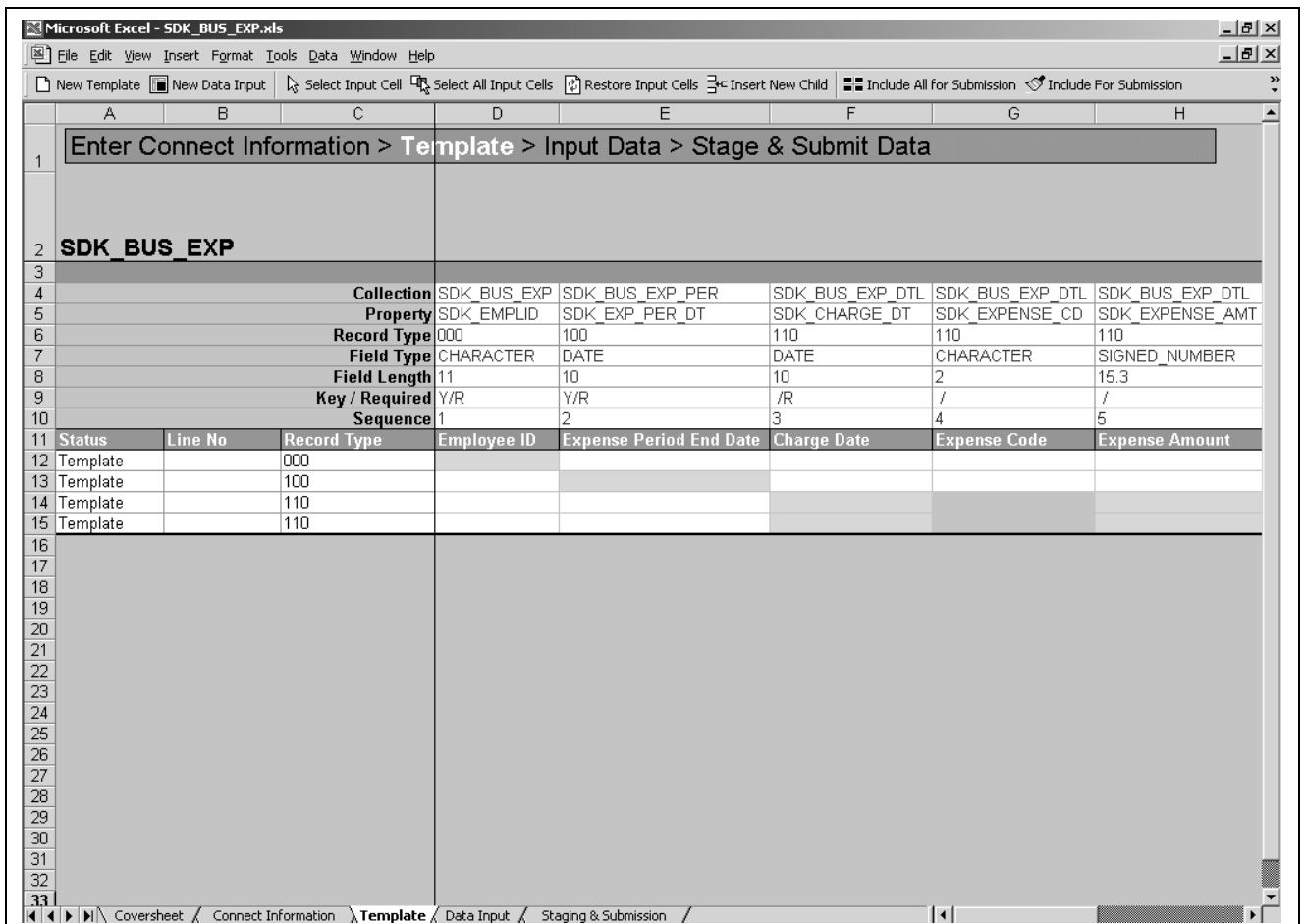
See [Chapter 9, “Using the Excel to Component Interface Utility,” Viewing Log Files, page 117.](#)

## Creating a Template

The Template page is a graphical representation of the component interface structure that you will be using to load data. The structure of the component interface is retrieved from the database when a new template is built. All of the fields that are exposed through the component interface appear on the template page. Fields that are read-only on the component interface will not appear on the template.

The New Template macro builds the parent-child relationship within Microsoft Excel based upon the component interface scroll level definition. The system adds a new row for each scroll level and assigns a unique identifier to it.

Access the Template tab in ExcelToCI.xls to create your template:



Template tab

### Collection

The name of the component interface collection. A collection is a property that points to a scroll, rather than a field, in the underlying component for a component interface.

### Property

The component interface property name. Typically, this is also the name of the field on the page.

### Record Type

This number represents the parent/child relationship of the records. The level 0 scroll record is always represented by 000. Level 1 scroll records appear with numbers that start with 100 and always have 00 as the last two numbers.

Level 2 scrolls are identified by numbers that start with the identifier of their level 1 parent and end with a 0.

Level 3 scrolls are identified by the first number from the level 1 parent, the second number from its level 2 parent, and then the third number from its own position in the list.

The numbers for each scroll level are incremented based on the number of records that exist at that level. For example, level 0 would be 000, level 1 would be 100, and level 2 would be 110, and so on.

---

**Note.** Component interfaces that have more than 10 collections at a given level will be incremented with alphabetic identifiers. For example, 800, 900, A00, and so on.

---

<b>Field Type</b>	The standard PeopleSoft type for the field. For example, <i>Date</i> , <i>Character</i> , and so on.
<b>Field Length</b>	The length of the field as defined by PeopleSoft. For numeric fields and signed number fields, the length is broken down into integer and decimal positions. For example, a length of 15.3 indicates 15 integer positions and three decimal positions.
<b>Key/Required</b>	If the field is a key field, the system will display a <i>Y</i> to the left of the forward slash. When the field is not a key, it will be blank. If the field is a required field, the system will display an <i>R</i> to the right of the forward slash. When the field is not required, it will be blank. This information comes from the record definition itself.
	<hr/> <b>Note.</b> Fields that are either keys or required <i>must</i> be set in order to submit data successfully. <hr/>
<b>Sequence</b>	The sequence number represents the property order in the template.
<b>Status</b>	This field displays the load status on the Staging and Submission page.
<b>Line No (number)</b>	This corresponds to the line number on the Input Data and the Staging and Submission pages.

## Understanding the Template Actions Toolbar

The Template Actions toolbar is made up of buttons that you use to create and modify a template, as well as create a Data Input sheet. You can resize the toolbar and move it to any location on the page or even drag it onto the existing standard Microsoft Excel toolbar. Once you have moved the toolbar to a location, it will remain there until you move it again. You do not need to move the toolbar each time you open the workbook.

Each button on the toolbar has help text that describes the purpose and use of each of the buttons when you place the cursor over the button. .

**New Template** Builds a new template based upon a component interface. The New Template macro builds the parent-child relationship within Microsoft Excel based upon the component interface structure.

When you build a new template, the system prompts you for your login information.

<b>New Data Input</b>	Builds a new Data Input sheet based upon the selected input cells. When you build a new Data Input sheet, the system prompts you as to whether you want to overwrite the existing sheet. If you select <i>Yes</i> a new Data Input sheet is created, overwriting the former one.
<b>Select Input Cell</b>	Selects an individual cell to be included in the Data Input Sheet. Cells that have been selected as input cells are highlighted in pink.
<b>Select All Input Cells</b>	Selects all properties to be included in the Data Input as input cells. When a cell is selected as an input cell, it is highlighted in pink.
<b>Restore Input Cells</b>	Restores the template to its original state and clears default values. The fields in the template will be highlighted in gray, indicating that nothing is included for submission.
<b>Insert New Child</b>	<p>Copies the selected row to be inserted as a new child. This creates multiple occurrences of the same record type.</p> <p>For example, if the selected row has a template identifier of 100, a new row is inserted that also has an identifier of 100 and is an exact duplicate of the selected row.</p> <hr/> <p><b>Note.</b> Use Insert New Child when multiple children must be submitted under the same parent record. Multiple children should not be created at identifier 000.</p> <hr/>
<b>Include All for Submission</b>	Includes all properties on the spreadsheet to be included for submission to the database. Cells that are included for submission appear only on the Staging and Submission sheet and do not appear on the Data Input sheet. Properties that are included for submission are highlighted in blue.
<b>Include for Submission</b>	Includes a single property to be included on the Staging and Submission sheet. Properties that use default values from the template must be included for submission. Cells that are included for submission generally are properties that contain default values or properties that you would like to see in the structure of the Staging and Submission sheet. Properties that are included for submission are highlighted in blue.
<b>Deselect Input Cell</b>	Changes a cell that was previously selected as an input cell to a cell that is included for submission. The cell is no longer included on the Data Input sheet but appears as part of the structure on the Staging and Submission sheet.
<b>Clear Template</b>	Clears all the data and structures on this sheet.
<b>Do Not Include for Submission</b>	Does not include the selected property for submission to the database. If a property is not included for submission, it will not appear in the structure that is submitted to the database on the Staging and Submission sheet. Properties that are not included for submission will only appear on the template worksheet and are not submitted to the database. Properties that are not included for submission are highlighted in gray.

---

**Note.** When you create a new template or a new Data Input sheet, the system clears the existing worksheet of all existing information. If you have a template or Data Input sheet that you need to save from previous uploads, save a copy of the worksheet before you create a new template or Data Input sheet.

---

## Entering Data into the Template

When determining which properties to include as input cells and which properties to include for submission, remember that the component interface uses the same business logic and executes the same PeopleCode as if the record were entered online using the page in your PeopleSoft application. To provide the minimal data necessary, these fields must either be provided with default (hard-coded) values or values that you provide using the Data Input sheet.

**Note.** You should unit test the template that you created with a few sample entries, and then verify your results before using the interface for mass input. For example, if you forgot to select a property, you will need to build a new Data Input sheet. If the results of the submission are satisfactory, continue entering data.

### Adding a New Child Record

By default, each collection is represented once on the template. To insert copies of a given collection, select that collection and click the Insert New Child button to create a copy of the selected row. The collection that you selected is copied so that you can have two rows under the same parent.

Enter Connect Information > Template > Input Data > Stage & Submit Data								
SDK_BUS_EXP								
Collection	SDK_BUS_EXP	SDK_BUS_EXP_PER	SDK_BUS_EXP_DTL	SDK_BUS_EXP_DTL	SDK_BUS_EXP_DTL	SDK_BUS_EXP_DTL	SDK_BUS_EXP	SDK_BUS_EXP
Property	SDK_EMPLID	SDK_EXP_PER_DT	SDK_CHARGE_DT	SDK_EXPENSE_CD	SDK_EXPENSE_AMT	SDK_CURRENC		
Record Type	000	100	110	110	110	110		
Field Type	CHARACTER	DATE	DATE	CHARACTER	SIGNED_NUMBER	CHARACTER		
Field Length	11	10	10	2	15.3	3		
Key / Required	Y/R	Y/R	/R	/	/	/R		
Sequence	1	2	3	4	5	6		
Status	Line No	Record Type	Employee ID	Expense Period End Date	Charge Date	Expense Code	Expense Amount	Currency Code
Template		000						
Template		100						
Template		110						
Template		110						

Template tab – child row added

**Note.** On the Data Input sheet (when the hierarchy is flattened) you will see duplicate columns where multiple children exist.

### Adding Default Values

Some fields have default values associated with them, either in the record definition or at runtime when the record is created on the database. Additionally, many components trigger PeopleCode, which populates default values, as well. To accept the database default, include the property for submission and the system default will be used.

There may be some fields for which you want to create your own default. For example, if you want to set the value of a field named *Status as of Effective Date* to A for every row that you submit, enter that value for the field in the template. Then, include the cell for submission on the template. The field will not appear on the Data Input page, but the value will appear in the field on the Submit to Database page. This is useful for effective dates, status fields, set IDs for simple imports, and so on.

When providing values for translate fields or prompt tables, provide the field value rather than the short or long description for the translate value. If you are unsure of the field values, check in the record or field definition in PeopleSoft Application Designer.

## Template Wrapping

If you are creating a template that has more than 252 properties, the Excel to Component Interface utility will wrap the additional columns and display them at the bottom of the spreadsheet so that they are still visible and can still be selected and deselected for submission.

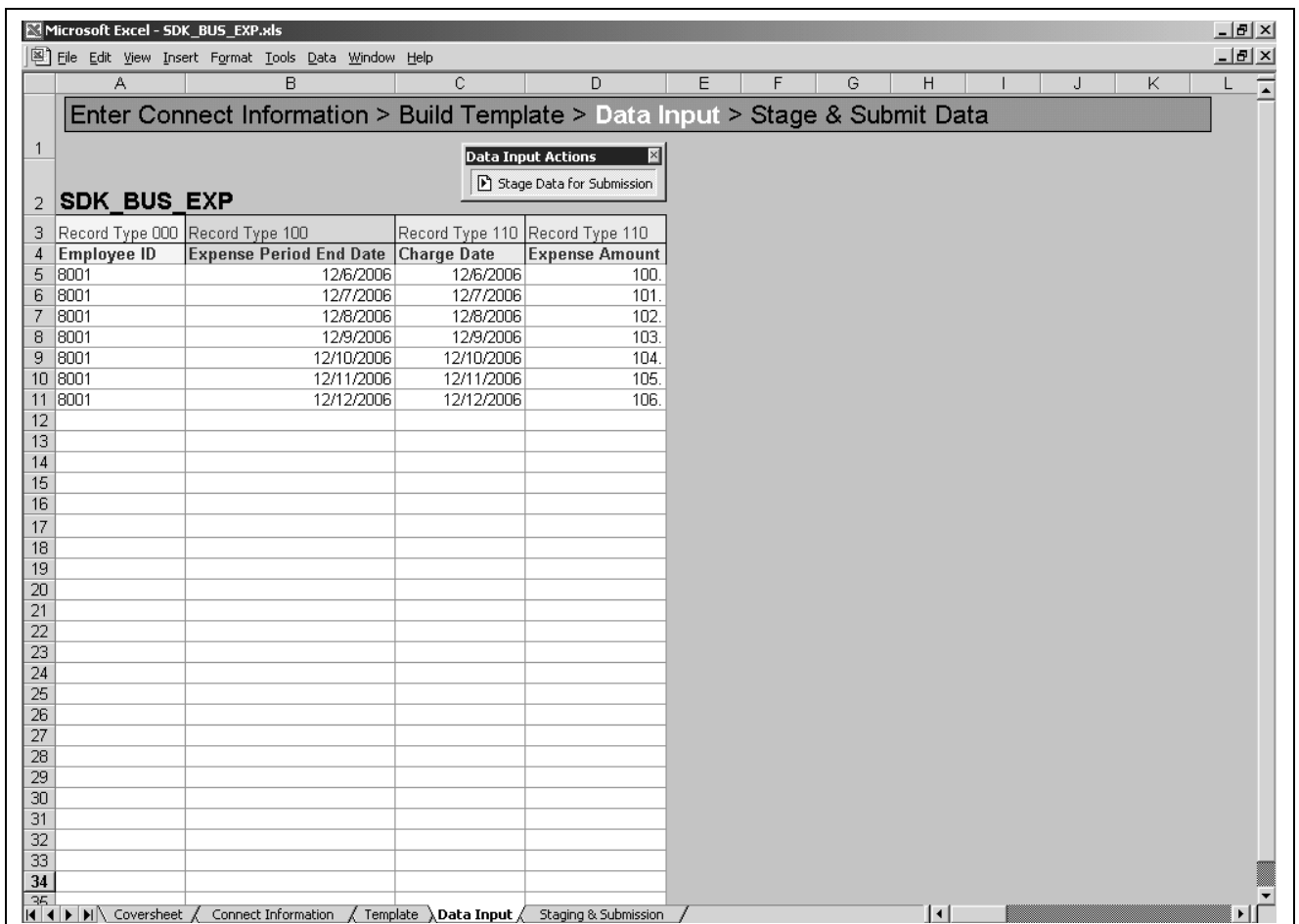
**Note.** Because of Microsoft Excel column limitations, no more than 252 columns may be submitted to the database at one time. When a template has more than 252 properties, you can work around this limitation in Microsoft Excel by deselecting an appropriate number of columns to meet the restriction before submitting to the database. You can determine in advance which fields should be selected as input cells, which cells should be included for submission, and which cells need not be included at all.

## Entering Data on the Data Input Sheet

The Data Input sheet enables you to enter data into the Excel to Component Interface utility so that it can be loaded to the database by using the component interface that you've selected. You can enter data manually or you can cut and paste it from another spreadsheet or third-party application.

## Using the Data Input Sheet

Access the Data Input tab to enter data:



Data Input tab

The field labels that appear on the Data Input sheet are those properties that you selected as input cells on the template. Each scroll level is identified by color. The record type from the template is also displayed for each property.

The system creates default date, datetime, and number formats when it creates the template. You can modify this format by using Microsoft Excel's default cell formatting when entering data, with the exception of the d/m/yy format for dates and datetimes. Instead, always use a d/m/yyyy format for dates and datetimes. To access the formatting feature select Format, Cells from the Excel menu.

The Data Input sheet is also used to correct data that failed to submit to the database. Errors that are flagged on the Submit to Database page are posted to the Data Input page, and when you have corrected them, the items marked in error can be staged again to the Staging and Submission sheet.

### **Data Input Actions**

The Data Input Actions toolbar contains the Stage Data for Submission button, which takes the data that you entered on the Data Input sheet and stages it for submission to the database. Once the data is staged, it appears on the Staging and Submission sheet in the hierarchical template structure. At this point, you should check that all fields are populated as expected. When the data is staged, it displays both the data on the Data Input sheet and the data that you specified as default values.

---

## **Viewing the Staged Data**

Access the Staging and Submission tab:

Microsoft Excel - SDK\_BUS\_EXP.xls

File Edit View Insert Format Tools Data Window Help

Enter Connect Information > Build Template > Input Data > Staging & Submission

10	Status	Line No	Record Type	Employee ID	Expense Period End Date	Charge Date	Expense Amount	Currency Code	
11		5	000	8001					
12		5	100		12/6/2006				
13		5	110			12/6/2006	100.	USD	
14		6	000	8001					
15		6	100		12/7/2006				
16		6	110			12/7/2006	101.	USD	
17		7	000	8001					
18		7	100		12/8/2006				
19		7	110			12/8/2006	102.	USD	
20		8	000	8001					
21		8	100		12/9/2006				
22		8	110			12/9/2006	103.	USD	
23		9	000	8001					
24		9	100		12/10/2006				
25		9	110			12/10/2006	104.	USD	
26		10	000	8001					
27		10	100		12/11/2006				
28		10	110			12/11/2006	105.	USD	
29		11	000	8001					
30		11	100		12/12/2006				
31		11	110			12/12/2006	106.	USD	
32	STOP								
33									
34									
35									
36									
37									
38									
39									
40									
41									
42									
43									

Staging & Submission Actions: Post Results, Submit Data

Navigation: Coversheet / Connect Information / Template / Data Input / Staging & Submission /

Staging and Submission tab

**Note.** Microsoft Excel has a physical limitation of 252 columns and 65k rows. Depending on the complexity of the record structure and the number of scroll-level records that are included in the template, you can use multiple rows on the Staging and Submission sheet to reflect one row of data from the Data Input sheet. To work around Microsoft Excel's limitations, you may need to stage your data in batches so that the number of rows on the Staging and Submission sheet will not exceed 65k.

## Staging and Submission Actions Toolbar

### Post Results

The results of the submission are copied to the Data Input sheet, where you can view the status of each row that is submitted and make any necessary corrections to rows that have the status of *Error*.

### Submit Data

The Login dialog box appears and you must specify your user ID and password.

The system submits the data to the database in the chunks that you specified on the Connection Information sheet.

After correcting any errors on the Data Input sheet, you can submit the data again. The items that had been marked as *Error* will be resubmitted, whereas those marked *OK* and *Warning* will be ignored.

## Error When Submitting Existing Keys

If you receive the error *Row already exists with the specified keys* and you are in CREATE mode, this indicates that the key already exists at level 0 or is part of the search record.

To verify that the key exists:

1. Open the component interface in PeopleSoft Application Designer.
2. Launch the component interface tester by selecting Tools, Test Component Interface.  
The Component Interface Tester search dialog box appears. This dialog box displays the keys (in the left-hand columns) for getting, creating, or finding an instance of the component interface.
3. Enter the value for the key you are testing.
4. Click Get Existing for the key that you are about to add, using the Excel to Component Interface utility.

If the Get Keys command returns the key, the key already exists and you must add data by using UPDATE mode.

If you receive a message that no row exists for the key, then the key does not exist at level 0 and the data should be added by using CREATE mode.

See [Chapter 3, “Developing the Component Interface,” Testing the Component Interface, page 44.](#)

## Correcting and Resubmitting Data

After you submit the data to the database, results of the process appear on the Staging and Submission sheet. If a submission had an error, the errored status appears on the Staging and Submission sheet. Use the Data Input page to correct the data and then resubmit it to the database. Continue this process of correcting errors and resubmitting until there are no errored records on the Data Input page.

---

**Note.** Data that was not submitted because the error threshold was reached will have no status. When the data that created the error is corrected on the Data Input sheet, the data that was not submitted will be staged to the database.

---

## Submission Statuses

Errors received for that record appear in a comment field by moving the cursor over the error flag. The records marked *OK* in green cannot be restaged for submission and can be kept as a record of work completed.

There are three statuses that can appear when you have submitted you data to the database.

<b><i>Ok</i></b>	The submission to the database completed successfully. The field will be highlighted in green.
<b><i>Warning</i></b>	The data was submitted to the database successfully, but a warning was generated in the process. The field will be highlighted in yellow.
<b><i>Error</i></b>	The data was not saved to the database due to an error. This field will be highlighted in red.  To see the error message that the component interface generated, place your cursor over the Status field. This message indicates how the data needs to be corrected.

## Creating a SOAP/XML Request

You can construct a SOAP/XML (Simple Object Access Protocol/Extensible Markup Language) request to create, update, or get component interface rows. The request and response contain component interface data in a SOAP/XML format.

### Request Format

The following example shows the request format:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" =>
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <Action__CompIntfc__CIName>
      Tags and Data
    </Action__CompIntfc__CIName>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Valid actions are Create, Get, Update, and Updatedata.

Ciname is the name of the component interface.

Tags and Data contains the tags and data for the component interface row or rows.

### Sample Create Request

The followings is a sample Create request:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <CREATE__CompIntfc__SUPPORT_DOC_TBL>
      <SUPPORT_DOC_ID>POLICE</SUPPORT_DOC_ID>
      <SUPPORT_DOC>
        <DESCR>Police Report</DESCR>
        <DESCRSHORT>Police</DESCRSHORT>
      </SUPPORT_DOC>
    </CREATE__CompIntfc__SUPPORT_DOC_TBL>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Sample Get Request

The following is a sample Get request:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
```

```

    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <Get__CompIntfc__SDK_BUS_EXP>
    <SDK_EMPLID>8052</SDK_EMPLID>
  </Get__CompIntfc__SDK_BUS_EXP>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Sample Update Request

The following is a sample Update request:

```

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <UPDATE__CompIntfc__SDK_BUS_EXP>
    <SDK_EMPLID>8001</SDK_EMPLID>
    <SDK_BUS_EXP_PER>
      <SDK_EXP_PER_DT>08/14/2002</SDK_EXP_PER_DT>
      <SDK_BUS_EXP_DTL>
        <SDK_CHARGE_DT>08/14/2002</SDK_CHARGE_DT>
        <SDK_EXPENSE_CD>01</SDK_EXPENSE_CD>
        <SDK_EXPENSE_AMT>1234.56</SDK_EXPENSE_AMT>
        <SDK_CURRENCY_CD>USD</SDK_CURRENCY_CD>
        <SDK_BUS_PURPOSE>Client Visit</SDK_BUS_PURPOSE>
        <SDK_DEPTID>10100</SDK_DEPTID>
      </SDK_BUS_EXP_DTL>
    </SDK_BUS_EXP_PER>
  </UPDATE__CompIntfc__SDK_BUS_EXP>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Sample Updatedata Request

The following is a sample Updatedata request:

```

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <UPDATEDATA__CompIntfc__USER_PROFILE>
      <UserID>VP1</UserID>
      <UserDescription>updated description</UserDescription>
      <EmailAddresses>
        <EmailType>BUS</EmailType>
        <EmailAddress>Updated@updated.com</EmailAddress>
      </EmailAddresses>
    </UPDATEDATA__CompIntfc__USER_PROFILE>

```

```

    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```

---

## Sending the Request

To send the request, post the XML code to the URL of the PeopleSoft Pure Internet Architecture server with the appropriate path to the iScript on the server.

---

**Note.** The PeopleSoft user ID and password must be sent in the SOAP request header with the identifiers of *userid* and *pwd*. You should send the request on a secure site.

---

Use this format:

```

Protocol(http or https)>://<WebServerMachineName>:<HTTPPort>/psc/ps/<Portal>=>
/<Node>/s/
WEBLIB_SOAPTOCI.SOAPTOCI.FieldFormula.IScript_SOAPToCI?&disconnect=y&postDataBin=y

```

<b>WebServerMachineName</b>	Machine name of the server.
<b>HTTPPort</b>	Port of the server.
<b>Portal</b>	Portal defined on the PIA server.
<b>Node</b>	Node defined on the PIA server.

For example,

```

http://MyWebServer:80/psc/ps/EMPLOYEE/PT_LOCAL/s/WEBLIB_
SOAPTOCI.SOAPTOCI.FieldFormula.IScript_SOAPToCI?disconnect=y&postDataBin=y

```

---

## Receiving a Response

This section provides examples of response types.

### Viewing a Response if a Row Already Exists

This is one example of the error response. The messages vary depending on the type of error.

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <USER_PROFILE>
      <Error-Warning>
        <Message>
          <Type>Error</Type>
          <MessageSetNumber>91</MessageSetNumber>
          <MessageNumber>49</MessageNumber>
          <MessageText>Row already exists with the specified keys.

```

```

        {USER_PROFILE} (91,49)</MessageText>
        <ExplainText>A rows already exists in the database with the specified⇒
keys.
        </ExplainText>
    </Message>
</Error-Warning>
<Key_information>
    <UserID>PTDM010</UserID>
</Key_information>
</USER_PROFILE>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Viewing a Sample Get Request and Response

The following XML code gets an SDK\_BUS\_EXP component interface row for an employee with an employee ID of 8052:

```

<?xml version="1.0"?>
<SDK_BUS_EXP action="GET">
    <SDK_EMPLID key="Y">8052</SDK_EMPLID>
    </SDK_BUS_EXP>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

The XML response for this employee is:

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <SDK_BUS_EXP>
            <SDK_BUS_EXP_PER>
                <SDK_EMPLID>8052</SDK_EMPLID>
                <SDK_EXP_PER_DT>2000-11-09</SDK_EXP_PER_DT>
                <SDK_BUS_EXP_DTL>
                    <SDK_EMPLID>8052</SDK_EMPLID>
                    <SDK_EXP_PER_DT>2000-11-09</SDK_EXP_PER_DT>
                    <SDK_CHARGE_DT />
                    <SDK_EXPENSE_CD />
                    <SDK_EXPENSE_AMT>0</SDK_EXPENSE_AMT>
                    <SDK_CURRENCY_CD>USD</SDK_CURRENCY_CD>
                    <SDK_BUS_PURPOSE />
                    <SDK_DEPTID />
                </SDK_BUS_EXP_DTL>
            </SDK_BUS_EXP_PER>
        </SDK_BUS_EXP>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Diagnosing and Resolving Errors

This section discusses how to:

- View the log files.
- Resolve error messages.

### Viewing Log Files

If you select the check box to create log files when building a template or submitting to the database, two log files are created—one that logs the activity of ExcelToCI.xls and the other that logs the SOAPTOCI Web Library.

The log for ExcelToCI.xls is created in the temp directory on the workstation running the Excel spreadsheet.

The log for the Web Library is created on the application server in the <PS\_HOME>\appserv\<DOMAIN>\files directory. This file contains both the SOAP request and the SOAP response.

Log files are written for each chunk of data that is sent to the database.

### Resolving Error Messages for Client Environments

Error Message	Possible Resolution
Component not correctly registered	Reinstall Visual Basic 6.0 SP5: Run-Time Redistribution Pack found on the Microsoft download site.
ActiveX component not correctly registered (Error 336)	Reinstall Visual Basic 6.0 SP5: Run-Time Redistribution Pack found at the Microsoft downloads website.
Error Number: -2147024770 Description: Automation error. The specified module could not be found.	<ul style="list-style-type: none"> <li>• Open Windows Explorer.</li> <li>• Navigate to c:\winnt\system32 directory and locate msxml4.dll.</li> <li>• Right-click the DLL and select Register COM Server. The message "DLLRegisterServer in c:\winnt\system32\msxml4.dll succeeded." will pop up.</li> <li>• Click OK.</li> </ul>
Error Number: 429 Description: ActiveX component can't create object.	<ul style="list-style-type: none"> <li>• Open Windows Explorer.</li> <li>• Navigate to c:\winnt\system32 directory and locate msxml4.dll.</li> <li>• Right-click the DLL and select Register COM Server. The message "DLLRegisterServer in c:\winnt\system32\msxml4.dll succeeded." will pop up.</li> <li>• Click OK.</li> </ul>

Error Message	Possible Resolution
Error Number -214722099 Description: Automation error in the dll	<ul style="list-style-type: none"> <li>• Open ExcelToCI.xls.</li> <li>• Press Alt + F11 to open the Microsoft Visual Basic Editor.</li> <li>• Select Tools, Add references.</li> <li>• Deselect anything that begins with Microsoft XML.</li> <li>• Browse for c:\winnt\system32msxml4.dll and click OK.</li> <li>• Select that version of msxml and click OK.</li> <li>• Click Save.</li> </ul>
Not Authorized (90,6)	The user trying to access this CI from ExcelToCI does not have access to the CI. Please provide access using PeopleTools > Security.

## CHAPTER 10

# Using WSDL Binding for Component Interfaces

This chapter provides an overview of the PeopleSoft implementation of the WSDL (Web Services Description Language) specification for the set of SOAP (Simple Object Access Protocol) transactions exposed for component interfaces, and covers:

- Setting up Integration Broker.
- Creating the third-party message node.
- Adding security to PeopleSoft objects.

---

**Note.** This chapter assumes that you are familiar with web architecture concepts, XML, web services, WSDL, and SOAP.

---

### See Also

<http://www.w3.org/TR/wsdl12-bindings>

---

## Understanding WSDL and Component Interfaces

SOAP Version 1.2 is a simple protocol used for exchanging structured information in a distributed network. WSDL Version 1.2 defines the XML grammar for describing network services as collections of communication endpoints capable of exchanging messages.

WSDL service definitions provide documentation for distributed systems and serve as a model for automating the processes involved in applications communication.

You can use both of these technologies with component interfaces.

---

**Note.** If a component interface has no authorized methods, WSDL will not be generated for it.

---

### See Also

*Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker*, “Creating Third-Party Integrations Using WSDL,” Generating WSDL from Component Interfaces and Messages

---

## Setting Up Integration Broker

To set up Integration Broker for use with WSDL and component interfaces, you’ll need to do the following:

- Ensure that the SOAPTOCI message is active.

- Verify the Message Channel.
- Modify gateway properties.
- Configure the local gateway.

## Ensuring That the SOAPTOCI Message Is Active

To check that the message is active:

1. Open PeopleSoft Application Designer.
2. Select File, Open.
3. In the Definition field, select *Message* and enter *SOAPTOCI* for the Name.
4. Click Open.
5. Select File, Definition Properties, select the Use tab, and make sure that the Status Active check box is selected.

## Verifying the Message Channel

To verify that *IB\_CHNL* is running:

1. Open the PeopleSoft Application Designer.
2. Select File, Open.
3. Select *Message Channel* from the the Definition field.
4. Enter *IB\_CHNL* for the name and click OK.
5. Select File, Definition Properties.
6. Check that Message Channel Status is set to Run on the Use tab.

## Modifying the Local Gateway

You must configure the properties used by the gateway to connect to the application server. To do this, you must have a valid PeopleSoft user ID and password.

The SOAP document does not specify a destination node; therefore, the gateway must forward the message to the default server. Each PeopleSoft environment that uses WSDL binding for the component interface must have its own Integration Broker running. Enter the following local default server information in this file.

### Configuring the Local Gateway

To modify the local gateway properties:

1. Open the PeopleSoft Internet Architecture.
2. Select PeopleTools, Integration Broker, Gateways.
3. Open the Gateway ID named LOCAL.
4. Enter the gateway URL: *http://server\_name:port/PSIGW/PeopleSoftListeningConnector*.
5. Click Save.
6. Click Load to load the connector information.
7. Click the Properties link.

You will be required to sign on to access the integrationGateway.properties file.

8. Use the Password Encryption utility on this page to generate an encrypted password.

This encrypted value is required to edit the integrationGateway.properties file.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker*, “Managing Integration Gateways,” Encrypting Passwords.

9. Edit the lines for ig.isc.\*

<b>ig.isc.serverURL</b>	Set Server URL.
<b>ig.isc.userid</b>	Set User ID.
<b>ig.isc.password</b>	Set User ID Password. Make sure to use the encrypted value of the password.
<b>ig.isc.toolsRel</b>	Set Tools Release.

For example,

```
ig.isc.serverURL=//MYSERVER:9000
ig.isc.userid=PTDMO
ig.isc.password= SQmYC3cuW0=
ig.isc.toolsRel=8.44
```

10. Click OK.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker*, “Managing Integration Gateways,” Using the integrationGateway.properties File.

## Creating the Third-Party Message Node

The third-party message node represents the system that will be making the SOAP requests to PeopleSoft.

To create a third-party message node:

1. Open the PeopleSoft Internet Architecture.
2. Select PeopleTools, Integration Broker, Node Definitions.
3. Select Add New Value.
4. Enter a Node Name
5. Click Add
6. Enter node information on the Node Info page.

See *Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker*, “Configuring Nodes and Transactions,” Configuring Nodes.

7. Click Save.
8. Select the Connectors tab.
9. Enter the following information:

<b>Gateway ID</b>	Set to LOCAL.
-------------------	---------------

<b>Connector ID</b>	Set to HTTPTARGET.
<b>Properties</b>	Enter a URL value for the PRIMARYURL property ID. Add two properties to the Default properties for HTTPTARGET Connector: <ul style="list-style-type: none"> <li>• Property ID: HEADER, Property Name: Content-Type, Value: text/xml</li> <li>• Property ID: HEADER, Property Name: SOAPAction, Value: "" (This property is needed only for SOAP communications from PeopleSoft to a third-party system.)</li> </ul>
<hr/>	
<b>Note.</b> This field grid does not appear until the Connector ID field is set to HTTPTARGET.	
<hr/>	

10. Click Save.

11. Select the Transactions tab.

12. Click Add Transactions.

Enter the following information:

**Transaction Type**                      Select Inbound Synchronous.

**Request Message**                      Select SOAPTOCI.

**Request Message Version**              Select VERSION\_1.

13. Click Add.

14. Select the Messages tab.

Enter the following in the Response Message group box:

**Name**                                      Set to SOAPTOCI.

**Version**                                    Set to VERSION\_1.

15. Click Save, then click Return to Transaction List.

16. Repeat the steps to create a Node Transaction with Transaction Type of OutboundSynchronous for the Request Message SOAPTOCI.

When you are finished with this procedure, you should have the following two transactions defined:

- Transaction Type: InSync, Request Message: SOAPTOCI.
- Transaction Type: OutSync, Request Message: SOAPTOCI.

---

## Adding Security to PeopleSoft Objects

This section covers:

- Adding security to a component interface.
- Adding security for SOAPTOCI web library.
- Adding security access for WSDL discovery.

## Adding Security to a Component Interface

Make sure to check the following items:

- Verify that the component interface you are using is listed; if not, add it.
- Verify that the method access is set appropriately.

Method access should be set according to user requirements of a particular permission list.

---

**Note.** PeopleTools does not support custom methods for WSDL generation.

---

See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Setting Up Permission Lists”.

## Adding Security for SOAPTOCI Web Library

To verify permissions for SOAPTOCI web library:

1. Select PeopleTools, Security, Permission Lists, and select the appropriate permission list.
2. Select Web Libraries, and verify that WEBLIB\_SOAPTOCI web library is listed; if not, add it.
3. Enter WEBLIB\_SOAPTOCI in the Web Library Name field.
4. Verify that the Access Permissions are set to *Full Access* for the web library functions.

See *Enterprise PeopleTools 8.45 PeopleBook: Security Administration*, “Setting Up Permission Lists”.

## Adding Security Access for WSDL Discovery

To enable WSDL discovery, the user profile must have access to WEBLIB\_soaptoci.WEBLIB\_soaptoci.WSDLSUMMARY.FieldFormula.IScript\_WSDLDiscovery.

In addition, that user profile also needs to have security access for any of the component interfaces that are going to be discovered.

---

# Generating WSDL for a Component Interface

The two approaches for generating the WSDL for a component interface are:

- Using the search capabilities within PeopleSoft on the Web Services for a component interface.
- Using direct access to the iScript through a URL.

## Generating WSDL Using PeopleSoft Search Capabilities

Access the Published EIPs page:

**Published EIPs**

**Component Interface**

Component Interface

Web Service:

Description:

**Message**

Message

Node Name:

Message Name:

Web Services to CI		
WSDL	Name	Methods
1	WSDL_CURRENCY_CD_CI	Create,Find,Get,Update,UpdateData
2	WSDL_MOBILE_PREFS_CI	Find,Get,Update,UpdateData
3	WSDL_PROCESSREQUEST	Create,Find,Get
4	WSDL_PRTL_SS_CI	Get
5	WSDL_PSACTIVITYLOG	Create,Find,Get,Update,UpdateData
6	WSDL_PT_AMM_DUMMY	Create,Find,Get,Update,UpdateData
7	WSDL_QE_ABS_HIST	Find,Get,Update,UpdateData
8	WSDL_QE_AE_REQUEST	Create,Find,Get,Update,UpdateData
9	WSDL_QE_BUS_EXP	Find,Get,Update,UpdateData
10	WSDL_QE_CI_BUS_EXP	Find,Get,Update,UpdateData

Published EIPs page

To generate WSDL, using the PeopleSoft search capability:

1. Select PeopleTools, Integration Broker, Web Services, Published EIPs.
2. Select Component Interface.
3. Click Search to display a list of component interfaces in the system that the given user is authorized to use.  
You can also enter the component interface name in the Web Service field and click Search to display an individual component interface.
4. Click the WSDL link to open a new window displaying the text of the WSDL.
5. If you have access to the methods of the component interface, they appear under the Methods column.

**Note.** If your web service supports uploading .wsdl files, you can copy and paste the WSDL into a .wsdl file.

### Generating WSDL by Directly Accessing the iScript

To access the WSDL directly through a URL string, the user ID and password must be sent in the soap header with the tags *userid* and *pwd*. Use the following URL format:

```
http://servername:port/psc/ps/PORTAL_NAME/LOCAL_NODE/s/WEBLIB_SO
APTOCI.WSDLSUMMARY.FieldFormula.iScript_WSDLDiscovery?&disconnect=y&postDataBin=&=&
y&APIName=
```

CI\_NAME

For example:

```
http://MyServer:80/psc/ps/EMPLOYEE/PT_LOCAL/s/WEBLIB_SOAPTOCI.WSDLSUMMAR
Y.FieldFormula.IScript_WSDLDiscovery?&disconne
ct=y&postDataBin=y&APIName=SDK_BUS_EXP
```

---

**Note.** Generally, the value for minOccurs is 0, meaning the element is not required. Keys and record fields marked as required have minOccurs values of 1, meaning that the element must be present.

---

### See Also

*Enterprise PeopleTools 8.45 PeopleBook: PeopleSoft Integration Broker*, “Creating Third-Party Integrations Using WSDL,” Generating WSDL from Component Interfaces and Messages

---

## Sending a SOAP Message to a PeopleSoft Application

The following code is an example of the Find method for the DEPT component interface in CRM.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="
http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" SOAP-ENV:
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <Security_Request>
      <Username>String</Username>
      <Password>String</Password>
    </Security_Request>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <Find__CompIntfc__CIName >
      <Key1>String</Key1>
      <Key2>String</Key2>
    </Find__CompIntfc__CIName>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

---

**Note.** PeopleSoft does not support namespaces in the body of the SOAP document.

---

## Understanding SOAP Considerations

This section covers SOAP considerations.

## Username and Password Considerations

The username and password for the security request portion of the SOAP document cannot refer to the same user that is specified for the integration gateway user. This user must have proper security access to the component interface that you plan to execute, and access to all the SOAPTOCI web library functions.

## SOAPAction Header and URL

The WSDL specifies the URL and SOAPAction for sending the SOAP document to a PeopleSoft application. For reference they are shown here:

### SOAPAction Header

The SOAPAction HTTP header uses this form:

```
SOAPAction: #MessageName#RequestingNode
```

MessageName is the name of the incoming message and is always SOAPTOCI. RequestingNode is the name of the requesting node.

See [Chapter 10, “Using WSDL Binding for Component Interfaces,” Creating the Third-Party Message Node, page 121.](#)

### URL

The SOAP Document should be sent to the following URL:

```
http://servername:port/PSIGW/HttpListeningConnector
```

---

## Testing SOAP to Component Interface by Using Send Master

To test the SOAP document:

1. Verify the URL by entering it into your browser.

The URL form is:

```
http://server_name:port/PSIGW/HttpListeningConnector
```

2. Locate Send Master in PS\_HOME\webserver\- 3. Run the batch file StartSendMaster.bat to start SendMaster test program.
- 4. Start a new project, enter a project name (SOAPTOCI), then click OK.
- 5. Enter the HttpListeningConnector URL in the ServerURL field.

The URL form is:

```
http://server_name:port/PSIGW/HttpListeningConnector
```

6. Enter the following headers in the Headers field:

```
Content-Type:text/xml
SOAPAction:#SOAPTOCI#THIRDPARTYNODE
```

where #THIRDPARTYNODE represents the name of the external third-party node that you defined earlier.

7. Enter your SOAP request message in the Input Information field.

You can use a program such as XML Spy to generate the SOAP request based on the generated XML. However, you must remove the namespaces from the elements in the SOAP body.

8. Click Post to post the SOAP to PeopleSoft.
9. View the response in the Output Information box.

### See Also

*Enterprise PeopleTools 8.45 PeopleBook: Integration Testing Utilities and Tools*,  
 “Using the Send Master Utility”

---

## Viewing an Example of a Find SOAP Request

The following is an example of a Find SOAP request for the DEPT component interface:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="
http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" SOAP-ENV:
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <Security_Request>
      <Username>PTDM01</Username>
      <Password>PTDM01</Password>
    </Security_Request>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <Find__CompIntfc__SDK_BUS_EXP>
      < SDK_EMPLID >80</ SDK_EMPLID >
    </Find__CompIntfc__SDK_BUS_EXP>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response from PeopleSoft is:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="
http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <Find__CompIntfc__SDK_BUS_EXPResponse>
      <SDK_BUS_EXP>
        <SDK_EMPLID>8001</SDK_EMPLID>
        <SDK_NAME>Schumacher,Sim on</SDK_NAME>
        <SDK_LAST_NAME_SRCH>ASD</SDK_LAST_NAME_SRCH>
      </SDK_BUS_EXP>
      <SDK_BUS_EXP>
        <SDK_EMPLID>8052</SDK_EMPLID>
        <SDK_NAME>Avery,Joan</SDK_NAME>
        <SDK_LAST_NAME_SRCH>AVERY</SDK_LAST_NAME_SRCH>
      </SDK_BUS_EXP>
```

```
    </Find__CompIntfc__SDK_BUS_EXPResponse>  
  </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

# APPENDIX A

## ISO Country and Currency Codes

PeopleBooks use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

This appendix discusses:

- ISO country codes.
- ISO currency codes.

### See Also

“About These PeopleBooks Preface,” Typographical Conventions and Visual Cues

---

## ISO Country Codes

This table lists the ISO country codes that may appear as country identifiers in PeopleBooks:

ISO Country Code	Country Name
ABW	Aruba
AFG	Afghanistan
AGO	Angola
AIA	Anguilla
ALB	Albania
AND	Andorra
ANT	Netherlands Antilles
ARE	United Arab Emirates
ARG	Argentina
ARM	Armenia
ASM	American Samoa
ATA	Antarctica

ISO Country Code	Country Name
ATF	French Southern Territories
ATG	Antigua and Barbuda
AUS	Australia
AUT	Austria
AZE	Azerbaijan
BDI	Burundi
BEL	Belgium
BEN	Benin
BFA	Burkina Faso
BGD	Bangladesh
BGR	Bulgaria
BHR	Bahrain
BHS	Bahamas
BIH	Bosnia and Herzegovina
BLR	Belarus
BLZ	Belize
BMU	Bermuda
BOL	Bolivia
BRA	Brazil
BRB	Barbados
BRN	Brunei Darussalam
BTN	Bhutan
BVT	Bouvet Island
BWA	Botswana
CAF	Central African Republic
CAN	Canada
CCK	Cocos (Keeling) Islands

ISO Country Code	Country Name
CHE	Switzerland
CHL	Chile
CHN	China
CIV	Cote D'Ivoire
CMR	Cameroon
COD	Congo, The Democratic Republic
COG	Congo
COK	Cook Islands
COL	Colombia
COM	Comoros
CPV	Cape Verde
CRI	Costa Rica
CUB	Cuba
CXR	Christmas Island
CYM	Cayman Islands
CYP	Cyprus
CZE	Czech Republic
DEU	Germany
DJI	Djibouti
DMA	Dominica
DNK	Denmark
DOM	Dominican Republic
DZA	Algeria
ECU	Ecuador
EGY	Egypt
ERI	Eritrea
ESH	Western Sahara

ISO Country Code	Country Name
ESP	Spain
EST	Estonia
ETH	Ethiopia
FIN	Finland
FJI	Fiji
FLK	Falkland Islands (Malvinas)
FRA	France
FRO	Faroe Islands
FSM	Micronesia, Federated States
GAB	Gabon
GBR	United Kingdom
GEO	Georgia
GHA	Ghana
GIB	Gibraltar
GIN	Guinea
GLP	Guadeloupe
GMB	Gambia
GNB	Guinea-Bissau
GNQ	Equatorial Guinea
GRC	Greece
GRD	Grenada
GRL	Greenland
GTM	Guatemala
GUF	French Guiana
GUM	Guam
GUY	Guyana
GXA	GXA - GP Core Country

ISO Country Code	Country Name
GXB	GXB - GP Core Country
GXC	GXC - GP Core Country
GXD	GXD - GP Core Country
HKG	Hong Kong
HMD	Heard and McDonald Islands
HND	Honduras
HRV	Croatia
HTI	Haiti
HUN	Hungary
IDN	Indonesia
IND	India
IOT	British Indian Ocean Territory
IRL	Ireland
IRN	Iran (Islamic Republic Of)
IRQ	Iraq
ISL	Iceland
ISR	Israel
ITA	Italy
JAM	Jamaica
JOR	Jordan
JPN	Japan
KAZ	Kazakstan
KEN	Kenya
KGZ	Kyrgyzstan
KHM	Cambodia
KIR	Kiribati
KNA	Saint Kitts and Nevis

ISO Country Code	Country Name
KOR	Korea, Republic of
KWT	Kuwait
LAO	Lao People's Democratic Rep
LBN	Lebanon
LBR	Liberia
LBY	Libyan Arab Jamahiriya
LCA	Saint Lucia
LIE	Liechtenstein
LKA	Sri Lanka
LSO	Lesotho
LTU	Lithuania
LUX	Luxembourg
LVA	Latvia
MAC	Macao
MAR	Morocco
MCO	Monaco
MDA	Moldova, Republic of
MDG	Madagascar
MDV	Maldives
MEX	Mexico
MHL	Marshall Islands
MKD	Fmr Yugoslav Rep of Macedonia
MLI	Mali
MLT	Malta
MMR	Myanmar
MNG	Mongolia
MNP	Northern Mariana Islands

ISO Country Code	Country Name
MOZ	Mozambique
MRT	Mauritania
MSR	Montserrat
MTQ	Martinique
MUS	Mauritius
MWI	Malawi
MYS	Malaysia
MYT	Mayotte
NAM	Namibia
NCL	New Caledonia
NER	Niger
NFK	Norfolk Island
NGA	Nigeria
NIC	Nicaragua
NIU	Niue
NLD	Netherlands
NOR	Norway
NPL	Nepal
NRU	Nauru
NZL	New Zealand
OMN	Oman
PAK	Pakistan
PAN	Panama
PCN	Pitcairn
PER	Peru
PHL	Philippines
PLW	Palau

ISO Country Code	Country Name
PNG	Papua New Guinea
POL	Poland
PRI	Puerto Rico
PRK	Korea, Democratic People's Rep
PRT	Portugal
PRY	Paraguay
PSE	Palestinian Territory, Occupie
PYF	French Polynesia
QAT	Qatar
REU	Reunion
ROU	Romania
RUS	Russian Federation
RWA	Rwanda
SAU	Saudi Arabia
SDN	Sudan
SEN	Senegal
SGP	Singapore
SGS	Sth Georgia & Sth Sandwich Is
SHN	Saint Helena
SJM	Svalbard and Jan Mayen
SLB	Solomon Islands
SLE	Sierra Leone
SLV	El Salvador
SMR	San Marino
SOM	Somalia
SPM	Saint Pierre and Miquelon
STP	Sao Tome and Principe

ISO Country Code	Country Name
SUR	Suriname
SVK	Slovakia
SVN	Slovenia
SWE	Sweden
SWZ	Swaziland
SYC	Seychelles
SYR	Syrian Arab Republic
TCA	Turks and Caicos Islands
TCD	Chad
TGO	Togo
THA	Thailand
TJK	Tajikistan
TKL	Tokelau
TKM	Turkmenistan
TLS	East Timor
TON	Tonga
TTO	Trinidad and Tobago
TUN	Tunisia
TUR	Turkey
TUV	Tuvalu
TWN	Taiwan, Province of China
TZA	Tanzania, United Republic of
UGA	Uganda
UKR	Ukraine
UMI	US Minor Outlying Islands
URY	Uruguay
USA	United States

ISO Country Code	Country Name
UZB	Uzbekistan
VAT	Holy See (Vatican City State)
VCT	St Vincent and the Grenadines
VEN	Venezuela
VGB	Virgin Islands (British)
VIR	Virgin Islands (U.S.)
VNM	Viet Nam
VUT	Vanuatu
WLF	Wallis and Futuna Islands
WSM	Samoa
YEM	Yemen
YUG	Yugoslavia
ZAF	South Africa
ZMB	Zambia
ZWE	Zimbabwe

---

## ISO Currency Codes

This table lists the ISO country codes that may appear as currency identifiers in PeopleBooks:

ISO Currency Code	Description
ADP	Andorran Peseta
AED	United Arab Emirates Dirham
AFA	Afghani
AFN	Afghani
ALK	Old Lek
ALL	Lek
AMD	Armenian Dram

ISO Currency Code	Description
ANG	Netherlands Antilles Guilder
AOA	Kwanza
AOK	Kwanza
AON	New Kwanza
AOR	Kwanza Reajustado
ARA	Austral
ARP	Peso Argentino
ARS	Argentine Peso
ARY	Peso
ATS	Schilling
AUD	Australian Dollar
AWG	Aruban Guilder
AZM	Azerbaijani Manat
BAD	Dinar
BAM	Convertible Marks
BBD	Barbados Dollar
BDT	Taka
BEC	Convertible Franc
BEF	Belgian Franc
BEL	Financial Belgian Franc
BGJ	Lev A/52
BGK	Lev A/62
BGL	Lev
BGN	Bulgarian LEV
BHD	Bahraini Dinar
BIF	Burundi Franc
BMD	Bermudian Dollar

ISO Currency Code	Description
BND	Brunei Dollar
BOB	Boliviano
BOP	Peso
BOV	Mvdol
BRB	Cruzeiro
BRC	Cruzado
BRE	Cruzeiro
BRL	Brazilian Real
BRN	New Cruzado
BRR	Brazilian Real Dollar
BSD	Bahamian Dollar
BTN	Ngultrum
BUK	N/A
BWP	Pula
BYB	Belarussian Ruble
BYR	Belarussian Ruble
BZD	Belize Dollar
CAD	Canadian Dollar
CDF	Franc Congolais
CHF	Swiss Franc
CLF	Unidades de fomento
CLP	Chilean Peso
CNX	Peoples Bank Dollar
CNY	Yuan Renminbi
COP	Colombian Peso
CRC	Costa Rican Colon
CSD	Serbia Dinar

ISO Currency Code	Description
CSJ	Krona A/53
CSK	Koruna
CUP	Cuban Peso
CVE	Cape Verde Escudo
CYP	Cyprus Pound
CZK	Czech Koruna
DEM	Deutsche Mark
DJF	Djibouti Franc
DKK	Danish Krone
DOP	Dominican Peso
DZD	Algerian Dinar
ECS	Sucre
ECV	Unidad de Valor
EEK	Kroon
EGP	Egyptian Pound
EQE	Ekwele
ERN	Nakfa
ESA	Spanish Peseta
ESB	Convertible Peseta
ESP	Spanish Peseta
ETB	Ethiopian Birr
EUR	euro
FIM	Markka
FJD	Fiji Dollar
FKP	Falklands Isl. Pound
FRF	French Franc
GBP	Pound Sterling

ISO Currency Code	Description
GEK	Georgian Coupon
GEL	Lari
GHC	Cedi
GIP	Gibraltar Pound
GMD	Dalasi
GNE	Syli
GNF	Guinea Franc
GNS	Syli
GQE	Ekwele
GRD	Drachma
GTQ	Quetzal
GWE	Guinea Escudo
GWP	Guinea-Bissau Peso
GYD	Guyana Dollar
HKD	Hong Kong Dollar
HNL	Lempira
HRD	Dinar
HRK	Kuna
HTG	Gourde
HUF	Forint
IDR	Rupiah
IEP	Irish Pound
ILP	Pound
ILR	Old Shekel
ILS	New Israeli Sheqel
INR	Indian Rupee
IQD	Iraqi Dinar

ISO Currency Code	Description
IRR	Iranian Rial
ISJ	Old Krona
ISK	Iceland Krona
ITL	Italian Lira
JMD	Jamaican Dollar
JOD	Jordanian Dinar
JPY	Yen
KES	Kenyan Shilling
KGS	Som
KHR	Riel
KMF	Comoro Franc
KPW	North Korean Won
KRW	Won
KWD	Kuwaiti Dinar
KYD	Cayman Islands dollar
KZT	Tenge
LAJ	Kip Pot Pol
LAK	Kip
LBP	Lebanese Pound
LKR	Sri Lanka Rupee
LRD	Liberian Dollar
LSL	Loti
LSM	Maloti
LTL	Lithuanian Litas
LTT	Talonas
LUC	Convertib Franc
LUF	Luxembourg Franc

ISO Currency Code	Description
LUL	Financial Franc
LVL	Latvian Lats
LVR	Latvian Ruble
LYD	Libyan Dinar
MAD	Moroccan Dirham
MAF	Mali Franc
MDL	Moldovan Leu
MGF	Malagasy Franc
MKD	Denar
MLF	Mali Franc
MMK	Kyat
MNT	Tugrik
MOP	Pataca
MRO	Ouguiya
MTL	Maltese Lira
MTP	Maltese Pound
MUR	Mauritius Rupee
MVQ	Maldive Rupee
MVR	Rufiyaa
MWK	Malawian Kwacha
MXN	Mexican Peso
MXP	Mexican Peso
MXV	Mexican UDI
MYR	Malaysian Ringgit
MZE	Mozambique Escudo
MZM	Metical
NAD	Namibia Dollar

ISO Currency Code	Description
NGN	Naira
NIC	Cordoba
NIO	Cordoba Oro
NLG	Netherlands Guilder
NOK	Norwegian Krone
NPR	Nepalese Rupee
NZD	New Zealand Dollar
OMR	Rial Omani
PAB	Balboa
PEI	Inti
PEN	Nuevo Sol
PES	Sol
PGK	Kina
PHP	Philippine Peso
PKR	Pakistan Rupee
PLN	Zloty
PLZ	Zloty
PTE	Portuguese Escudo
PYG	Guarani
QAR	Qatari Rial
ROK	Leu A/52
ROL	Leu
RUB	Russian Ruble
RUR	Russian Federation Rouble
RWF	Rwanda Franc
SAR	Saudi Riyal
SBD	Solomon Islands

ISO Currency Code	Description
SCR	Seychelles Rupee
SDD	Sudanese Dinar
SDP	Sudanese Pound
SEK	Swedish Krona
SGD	Singapore Dollar
SHP	St Helena Pound
SIT	Tolar
SKK	Slovak Koruna
SLL	Leone
SOS	Somali Shilling
SRG	Surinam Guilder
STD	Dobra
SUR	Rouble
SVC	El Salvador Colon
SYP	Syrian Pound
SZL	Lilangeni
THB	Baht
TJR	Tajik Ruble
TJS	Somoni
TMM	Manat
TND	Tunisian Dinar
TOP	Pa'anga
TPE	Timor Escudo
TRL	Turkish Lira
TTD	Trinidad Dollar
TWD	New Taiwan Dollar
TZS	Tanzanian Shilling

ISO Currency Code	Description
UAH	Hryvnia
UAK	Karbovanet
UGS	Uganda Shilling
UGW	Old Shilling
UGX	Uganda Shilling
USD	US Dollar
USN	US Dollar (Next day)
USS	US Dollar (Same day)
UYN	Old Uruguay Peso
UYP	Uruguayan Peso
UYU	Peso Uruguayo
UZS	Uzbekistan Sum
VEB	Bolivar
VNC	Old Dong
VND	Dong
VUV	Vatu
WST	Tala
XAF	CFA Franc BEAC
XAG	Silver
XAU	GOLD
XBA	European Composite Unit
XBB	European Monetary Unit
XBC	European Unit of Account 9
XBD	European Unit of Account 17
XCD	East Caribbean Dollar
XDR	SDR
XEU	EU Currency (E.C.U)

ISO Currency Code	Description
XFO	Gold-Franc
XFU	UIC-Franc
XOF	CFA Franc BCEAO
XPD	Palladium
XPF	CFP Franc
XPT	Platinum
XTS	For Testing Purposes
XXX	Non Currency Transaction
YDD	Yemeni Din
YER	Yemeni Rial
YUD	New Yugoslavian Dinar
YUM	New Dinar
YUN	Yugoslavian Dinar
ZAL	Financial Rand
ZAR	Rand
ZMK	Zambian Kwacha
ZRN	New Zaire
ZRZ	Zaire
ZWC	Rhodesian Dollar
ZWD	Zimbabwe Dollar

# Glossary of PeopleSoft Terms

<b>absence entitlement</b>	This element defines rules for granting paid time off for valid absences, such as sick time, vacation, and maternity leave. An absence entitlement element defines the entitlement amount, frequency, and entitlement period.
<b>absence take</b>	This element defines the conditions that must be met before a payee is entitled to take paid time off.
<b>accounting class</b>	In PeopleSoft Enterprise Performance Management, the accounting class defines how a resource is treated for generally accepted accounting practices. The Inventory class indicates whether a resource becomes part of a balance sheet account, such as inventory or fixed assets, while the Non-inventory class indicates that the resource is treated as an expense of the period during which it occurs.
<b>accounting date</b>	The accounting date indicates when a transaction is recognized, as opposed to the date the transaction actually occurred. The accounting date and transaction date can be the same. The accounting date determines the period in the general ledger to which the transaction is to be posted. You can only select an accounting date that falls within an open period in the ledger to which you are posting. The accounting date for an item is normally the invoice date.
<b>accounting split</b>	The accounting split method indicates how expenses are allocated or divided among one or more sets of accounting ChartFields.
<b>accumulator</b>	You use an accumulator to store cumulative values of defined items as they are processed. You can accumulate a single value over time or multiple values over time. For example, an accumulator could consist of all voluntary deductions, or all company deductions, enabling you to accumulate amounts. It allows total flexibility for time periods and values accumulated.
<b>action reason</b>	The reason an employee's job or employment information is updated. The action reason is entered in two parts: a personnel action, such as a promotion, termination, or change from one pay group to another—and a reason for that action. Action reasons are used by PeopleSoft Human Resources, PeopleSoft Benefits Administration, PeopleSoft Stock Administration, and the COBRA Administration feature of the Base Benefits business process.
<b>action template</b>	In PeopleSoft Receivables, outlines a set of escalating actions that the system or user performs based on the period of time that a customer or item has been in an action plan for a specific condition.
<b>activity</b>	<p>In PeopleSoft Enterprise Learning Management, an instance of a catalog item (sometimes called a class) that is available for enrollment. The activity defines such things as the costs that are associated with the offering, enrollment limits and deadlines, and waitlisting capacities.</p> <p>In PeopleSoft Enterprise Performance Management, the work of an organization and the aggregation of actions that are used for activity-based costing.</p> <p>In PeopleSoft Project Costing, the unit of work that provides a further breakdown of projects—usually into specific tasks.</p> <p>In PeopleSoft Workflow, a specific transaction that you might need to perform in a business process. Because it consists of the steps that are used to perform a transaction, it is also known as a step map.</p>

<b>agreement</b>	In PeopleSoft eSettlements, provides a way to group and specify processing options, such as payment terms, pay from a bank, and notifications by a buyer and supplier location combination.
<b>allocation rule</b>	In PeopleSoft Enterprise Incentive Management, an expression within compensation plans that enables the system to assign transactions to nodes and participants. During transaction allocation, the allocation engine traverses the compensation structure from the current node to the root node, checking each node for plans that contain allocation rules.
<b>alternate account</b>	A feature in PeopleSoft General Ledger that enables you to create a statutory chart of accounts and enter statutory account transactions at the detail transaction level, as required for recording and reporting by some national governments.
<b>AR specialist</b>	Abbreviation for <i>receivables specialist</i> . In PeopleSoft Receivables, an individual in who tracks and resolves deductions and disputed items.
<b>arbitration plan</b>	In PeopleSoft Enterprise Pricer, defines how price rules are to be applied to the base price when the transaction is priced.
<b>assessment rule</b>	In PeopleSoft Receivables, a user-defined rule that the system uses to evaluate the condition of a customer's account or of individual items to determine whether to generate a follow-up action.
<b>asset class</b>	An asset group used for reporting purposes. It can be used in conjunction with the asset category to refine asset classification.
<b>attribute/value pair</b>	In PeopleSoft Directory Interface, relates the data that makes up an entry in the directory information tree.
<b>authentication server</b>	A server that is set up to verify users of the system.
<b>base time period</b>	In PeopleSoft Business Planning, the lowest level time period in a calendar.
<b>benchmark job</b>	In PeopleSoft Workforce Analytics, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.
<b>book</b>	In PeopleSoft Asset Management, used for storing financial and tax information, such as costs, depreciation attributes, and retirement information on assets.
<b>branch</b>	A tree node that rolls up to nodes above it in the hierarchy, as defined in PeopleSoft Tree Manager.
<b>budgetary account only</b>	An account used by the system only and not by users; this type of account does not accept transactions. You can only budget with this account. Formerly called "system-maintained account."
<b>budget check</b>	In commitment control, the processing of source transactions against control budget ledgers, to see if they pass, fail, or pass with a warning.
<b>budget control</b>	In commitment control, budget control ensures that commitments and expenditures don't exceed budgets. It enables you to track transactions against corresponding budgets and terminate a document's cycle if the defined budget conditions are not met. For example, you can prevent a purchase order from being dispatched to a vendor if there are insufficient funds in the related budget to support it.
<b>budget period</b>	The interval of time (such as 12 months or 4 quarters) into which a period is divided for budgetary and reporting purposes. The ChartField allows maximum flexibility to define operational accounting time periods without restriction to only one calendar.
<b>business event</b>	In PeopleSoft Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.

	In PeopleSoft Sales Incentive Management, an original business transaction or activity that may justify the creation of a PeopleSoft Enterprise Incentive Management event (a sale, for example).
<b>business unit</b>	A corporation or a subset of a corporation that is independent with regard to one or more operational or accounting functions.
<b>buyer</b>	In PeopleSoft eSettlements, an organization (or business unit, as opposed to an individual) that transacts with suppliers (vendors) within the system. A buyer creates payments for purchases that are made in the system.
<b>catalog item</b>	In PeopleSoft Enterprise Learning Management, a specific topic that a learner can study and have tracked. For example, "Introduction to Microsoft Word." A catalog item contains general information about the topic and includes a course code, description, categorization, keywords, and delivery methods. A catalog item can have one or more learning activities.
<b>catalog map</b>	In PeopleSoft Catalog Management, translates values from the catalog source data to the format of the company's catalog.
<b>catalog partner</b>	In PeopleSoft Catalog Management, shares responsibility with the enterprise catalog manager for maintaining catalog content.
<b>categorization</b>	Associates partner offerings with catalog offerings and groups them into enterprise catalog categories.
<b>channel</b>	In PeopleSoft MultiChannel Framework, email, chat, voice (computer telephone integration [CTI]), or a generic event.
<b>ChartField</b>	A field that stores a chart of accounts, resources, and so on, depending on the PeopleSoft application. ChartField values represent individual account numbers, department codes, and so forth.
<b>ChartField balancing</b>	You can require specific ChartFields to match up (balance) on the debit and the credit side of a transaction.
<b>ChartField combination edit</b>	The process of editing journal lines for valid ChartField combinations based on user-defined rules.
<b>ChartKey</b>	One or more fields that uniquely identify each row in a table. Some tables contain only one field as the key, while others require a combination.
<b>checkbook</b>	In PeopleSoft Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.
<b>Class ChartField</b>	A ChartField value that identifies a unique appropriation budget key when you combine it with a fund, department ID, and program code, as well as a budget period. Formerly called <i>sub-classification</i> .
<b>clone</b>	In PeopleCode, to make a unique copy. In contrast, to <i>copy</i> may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.
<b>collection</b>	To make a set of documents available for searching in Verity, you must first create at least one collection. A collection is set of directories and files that allow search application users to use the Verity search engine to quickly find and display source documents that match search criteria. A collection is a set of statistics and pointers to the source documents, stored in a proprietary format on a file server. Because a collection can only store information for a single location, PeopleSoft maintains a set of collections (one per language code) for each search index object.

<b>collection rule</b>	In PeopleSoft Receivables, a user-defined rule that defines actions to take for a customer based on both the amount and the number of days past due for outstanding balances.
<b>compensation object</b>	In PeopleSoft Enterprise Incentive Management, a node within a compensation structure. Compensation objects are the building blocks that make up a compensation structure's hierarchical representation.
<b>compensation structure</b>	In PeopleSoft Enterprise Incentive Management, a hierarchical relationship of compensation objects that represents the compensation-related relationship between the objects.
<b>condition</b>	In PeopleSoft Receivables, occurs when there is a change of status for a customer's account, such as reaching a credit limit or exceeding a user-defined balance due.
<b>configuration parameter catalog</b>	Used to configure an external system with PeopleSoft. For example, a configuration parameter catalog might set up configuration and communication parameters for an external server.
<b>configuration plan</b>	In PeopleSoft Enterprise Incentive Management, configuration plans hold allocation information for common variables (not incentive rules) and are attached to a node without a participant. Configuration plans are not processed by transactions.
<b>content reference</b>	Content references are pointers to content registered in the portal registry. These are typically either URLs or iScripts. Content references fall into three categories: target content, templates, and template pagelets.
<b>context</b>	In PeopleCode, determines which buffer fields can be contextually referenced and which is the current row of data on each scroll level when a PeopleCode program is running.  In PeopleSoft Enterprise Incentive Management, a mechanism that is used to determine the scope of a processing run. PeopleSoft Enterprise Incentive Management uses three types of context: plan, period, and run-level.
<b>control table</b>	Stores information that controls the processing of an application. This type of processing might be consistent throughout an organization, or it might be used only by portions of the organization for more limited sharing of data.
<b>cost profile</b>	A combination of a receipt cost method, a cost flow, and a deplete cost method. A profile is associated with a cost book and determines how items in that book are valued, as well as how the material movement of the item is valued for the book.
<b>cost row</b>	A cost transaction and amount for a set of ChartFields.
<b>current learning</b>	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's in-progress learning activities and programs.
<b>data acquisition</b>	In PeopleSoft Enterprise Incentive Management, the process during which raw business transactions are acquired from external source systems and fed into the operational data store (ODS).
<b>data elements</b>	Data elements, at their simplest level, define a subset of data and the rules by which to group them.  For Workforce Analytics, data elements are rules that tell the system what measures to retrieve about your workforce groups.
<b>dataset</b>	A data grouping that enables role-based filtering and distribution of data. You can limit the range and quantity of data that is displayed for a user by associating dataset rules with user roles. The result of dataset rules is a set of data that is appropriate for the user's roles.

<b>delivery method</b>	<p>In PeopleSoft Enterprise Learning Management, identifies the primary type of delivery method in which a particular learning activity is offered. Also provides default values for the learning activity, such as cost and language. This is primarily used to help learners search the catalog for the type of delivery from which they learn best. Because PeopleSoft Enterprise Learning Management is a blended learning system, it does not enforce the delivery method.</p> <p>In PeopleSoft Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, rail, and so on). The delivery method is specified when creating shipment schedules.</p>
<b>delivery method type</b>	In PeopleSoft Enterprise Learning Management, identifies how learning activities can be delivered—for example, through online learning, classroom instruction, seminars, books, and so forth—in an organization. The type determines whether the delivery method includes scheduled components.
<b>directory information tree</b>	In PeopleSoft Directory Interface, the representation of a directory’s hierarchical structure.
<b>document sequencing</b>	A flexible method that sequentially numbers the financial transactions (for example, bills, purchase orders, invoices, and payments) in the system for statutory reporting and for tracking commercial transaction activity.
<b>dynamic detail tree</b>	A tree that takes its detail values—dynamic details—directly from a table in the database, rather than from a range of values that are entered by the user.
<b>edit table</b>	A table in the database that has its own record definition, such as the Department table. As fields are entered into a PeopleSoft application, they can be validated against an edit table to ensure data integrity throughout the system.
<b>effective date</b>	A method of dating information in PeopleSoft applications. You can predate information to add historical data to your system, or postdate information in order to enter it before it actually goes into effect. By using effective dates, you don’t delete values; you enter a new value with a current effective date.
<b>EIM ledger</b>	Abbreviation for <i>Enterprise Incentive Management ledger</i> . In PeopleSoft Enterprise Incentive Management, an object to handle incremental result gathering within the scope of a participant. The ledger captures a result set with all of the appropriate traces to the data origin and to the processing steps of which it is a result.
<b>elimination set</b>	In PeopleSoft General Ledger, a related group of intercompany accounts that is processed during consolidations.
<b>entry event</b>	In PeopleSoft General Ledger, Receivables, Payables, Purchasing, and Billing, a business process that generates multiple debits and credits resulting from single transactions to produce standard, supplemental accounting entries.
<b>equitization</b>	In PeopleSoft General Ledger, a business process that enables parent companies to calculate the net income of subsidiaries on a monthly basis and adjust that amount to increase the investment amount and equity income amount before performing consolidations.
<b>event</b>	<p>A predefined point either in the Component Processor flow or in the program flow. As each point is encountered, the event activates each component, triggering any PeopleCode program that is associated with that component and that event. Examples of events are FieldChange, SavePreChange, and RowDelete.</p> <p>In PeopleSoft Human Resources, also refers to an incident that affects benefits eligibility.</p>
<b>event propagation process</b>	In PeopleSoft Sales Incentive Management, a process that determines, through logic, the propagation of an original PeopleSoft Enterprise Incentive Management event and creates a derivative (duplicate) of the original event to be processed by other objects.

	Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.
<b>exception</b>	In PeopleSoft Receivables, an item that either is a deduction or is in dispute.
<b>exclusive pricing</b>	In PeopleSoft Order Management, a type of arbitration plan that is associated with a price rule. Exclusive pricing is used to price sales order transactions.
<b>fact</b>	In PeopleSoft applications, facts are numeric data values from fields from a source database as well as an analytic application. A fact can be anything you want to measure your business by, for example, revenue, actual, budget data, or sales numbers. A fact is stored on a fact table.
<b>forecast item</b>	A logical entity with a unique set of descriptive demand and forecast data that is used as the basis to forecast demand. You create forecast items for a wide range of uses, but they ultimately represent things that you buy, sell, or use in your organization and for which you require a predictable usage.
<b>fund</b>	In PeopleSoft Promotions Management, a budget that can be used to fund promotional activity. There are four funding methods: top down, fixed accrual, rolling accrual, and zero-based accrual.
<b>generic process type</b>	In PeopleSoft Process Scheduler, process types are identified by a generic process type. For example, the generic process type SQR includes all SQR process types, such as SQR process and SQR report.
<b>group</b>	In PeopleSoft Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs).  In PeopleSoft Human Resources Management and Supply Chain Management, any set of records that are associated under a single name or variable to run calculations in PeopleSoft business processes. In PeopleSoft Time and Labor, for example, employees are placed in groups for time reporting purposes.
<b>incentive object</b>	In PeopleSoft Enterprise Incentive Management, the incentive-related objects that define and support the PeopleSoft Enterprise Incentive Management calculation process and results, such as plan templates, plans, results data, user interaction objects, and so on.
<b>incentive rule</b>	In PeopleSoft Sales Incentive Management, the commands that act on transactions and turn them into compensation. A rule is one part in the process of turning a transaction into compensation.
<b>incur</b>	In PeopleSoft Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.
<b>item</b>	In PeopleSoft Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse).  In PeopleSoft Demand Planning, Inventory Policy Planning, and Supply Planning, a noninventory item that is designated as being used for planning purposes only. It can represent a family or group of inventory items. It can have a planning bill of material (BOM) or planning routing, and it can exist as a component on a planning BOM. A planning item cannot be specified on a production or engineering BOM or routing, and it cannot be used as a component in a production. The quantity on hand will never be maintained.
	In PeopleSoft Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.
<b>KPI</b>	An abbreviation for <i>key performance indicator</i> . A high-level measurement of how well an organization is doing in achieving critical success factors. This defines the data value or calculation upon which an assessment is determined.

<b>LDIF file</b>	Abbreviation for <i>Lightweight Directory Access Protocol (LDAP) Data Interchange Format file</i> . Contains discrepancies between PeopleSoft data and directory data.
<b>learner group</b>	In PeopleSoft Enterprise Learning Management, a group of learners who are linked to the same learning environment. Members of the learner group can share the same attributes, such as the same department or job code. Learner groups are used to control access to and enrollment in learning activities and programs. They are also used to perform group enrollments and mass enrollments in the back office.
<b>learning components</b>	In PeopleSoft Enterprise Learning Management, the foundational building blocks of learning activities. PeopleSoft Enterprise Learning Management supports six basic types of learning components: web-based, session, webcast, test, survey, and assignment. One or more of these learning component types compose a single learning activity.
<b>learning environment</b>	In PeopleSoft Enterprise Learning Management, identifies a set of categories and catalog items that can be made available to learner groups. Also defines the default values that are assigned to the learning activities and programs that are created within a particular learning environment. Learning environments provide a way to partition the catalog so that learners see only those items that are relevant to them.
<b>learning history</b>	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's completed learning activities and programs.
<b>ledger mapping</b>	You use ledger mapping to relate expense data from general ledger accounts to resource objects. Multiple ledger line items can be mapped to one or more resource IDs. You can also use ledger mapping to map dollar amounts (referred to as <i>rates</i> ) to business units. You can map the amounts in two different ways: an actual amount that represents actual costs of the accounting period, or a budgeted amount that can be used to calculate the capacity rates as well as budgeted model results. In PeopleSoft Enterprise Warehouse, you can map general ledger accounts to the EW Ledger table.
<b>library section</b>	In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan (or template) and that is available for other plans to share. Changes to a library section are reflected in all plans that use it.
<b>linked section</b>	In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan template but appears in a plan. Changes to linked sections propagate to plans using that section.
<b>linked variable</b>	In PeopleSoft Enterprise Incentive Management, a variable that is defined and maintained in a plan template and that also appears in a plan. Changes to linked variables propagate to plans using that variable.
<b>load</b>	In PeopleSoft Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Inventory that is used to track the weight, the volume, and the destination of a shipment.
<b>local functionality</b>	In PeopleSoft HRMS, the set of information that is available for a specific country. You can access this information when you click the appropriate country flag in the global window, or when you access it by a local country menu.
<b>location</b>	Locations enable you to indicate the different types of addresses—for a company, for example, one address to receive bills, another for shipping, a third for postal deliveries, and a separate street address. Each address has a different location number. The primary location—indicated by a <i>1</i> —is the address you use most often and may be different from the main address.
<b>logistical task</b>	In PeopleSoft Services Procurement, an administrative task that is related to hiring a service provider. Logistical tasks are linked to the service type on the work order so that different types of services can have different logistical tasks. Logistical tasks include both preapproval tasks (such as assigning a new badge or ordering a new

laptop) and postapproval tasks (such as scheduling orientation or setting up the service provider email). The logistical tasks can be mandatory or optional. Mandatory preapproval tasks must be completed before the work order is approved. Mandatory postapproval tasks, on the other hand, must be completed before a work order is released to a service provider.

<b>market template</b>	In PeopleSoft Enterprise Incentive Management, additional functionality that is specific to a given market or industry and is built on top of a product category.
<b>match group</b>	In PeopleSoft Receivables, a group of receivables items and matching offset items. The system creates match groups by using user-defined matching criteria for selected field values.
<b>MCF server</b>	Abbreviation for <i>PeopleSoft MultiChannel Framework server</i> . Comprises the universal queue server and the MCF log server. Both processes are started when <i>MCF Servers</i> is selected in an application server domain configuration.
<b>merchandising activity</b>	In PeopleSoft Promotions Management, a specific discount type that is associated with a trade promotion (such as off-invoice, billback or rebate, or lump-sum payment) that defines the performance that is required to receive the discount. In the industry, you may know this as an offer, a discount, a merchandising event, an event, or a tactic.
<b>meta-SQL</b>	Meta-SQL constructs expand into platform-specific Structured Query Language (SQL) substrings. They are used in functions that pass SQL strings, such as in SQL objects, the SQLExec function, and PeopleSoft Application Engine programs.
<b>metastring</b>	Metastrings are special expressions included in SQL string literals. The metastrings, prefixed with a percent (%) symbol, are included directly in the string literals. They expand at run time into an appropriate substring for the current database platform.
<b>multibook</b>	In PeopleSoft General Ledger, multiple ledgers having multiple-base currencies that are defined for a business unit, with the option to post a single transaction to all base currencies (all ledgers) or to only one of those base currencies (ledgers).
<b>multicurrency</b>	The ability to process transactions in a currency other than the business unit's base currency.
<b>national allowance</b>	In PeopleSoft Promotions Management, a promotion at the corporate level that is funded by nondiscretionary dollars. In the industry, you may know this as a national promotion, a corporate promotion, or a corporate discount.
<b>node-oriented tree</b>	A tree that is based on a detail structure, but the detail values are not used.
<b>pagelet</b>	Each block of content on the home page is called a pagelet. These pagelets display summary information within a small rectangular area on the page. The pagelet provide users with a snapshot of their most relevant PeopleSoft and non-PeopleSoft content.
<b>participant</b>	In PeopleSoft Enterprise Incentive Management, participants are recipients of the incentive compensation calculation process.
<b>participant object</b>	Each participant object may be related to one or more compensation objects. See also <i>compensation object</i> .
<b>partner</b>	A company that supplies products or services that are resold or purchased by the enterprise.
<b>pay cycle</b>	In PeopleSoft Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.
<b>pending item</b>	In PeopleSoft Receivables, an individual receivable (such as an invoice, a credit memo, or a write-off) that has been entered in or created by the system, but hasn't been posted.

<b>PeopleCode</b>	PeopleCode is a proprietary language, executed by the PeopleSoft application processor. PeopleCode generates results based upon existing data or user actions. By using business interlink objects, external services are available to all PeopleSoft applications wherever PeopleCode can be executed.
<b>PeopleCode event</b>	An action that a user takes upon an object, usually a record field, that is referenced within a PeopleSoft page.
<b>PeopleSoft Internet Architecture</b>	The fundamental architecture on which PeopleSoft 8 applications are constructed, consisting of a relational database management system (RDBMS), an application server, a web server, and a browser.
<b>performance measurement</b>	In PeopleSoft Enterprise Incentive Management, a variable used to store data (similar to an aggregator, but without a predefined formula) within the scope of an incentive plan. Performance measures are associated with a plan calendar, territory, and participant. Performance measurements are used for quota calculation and reporting.
<b>period context</b>	In PeopleSoft Enterprise Incentive Management, because a participant typically uses the same compensation plan for multiple periods, the period context associates a plan context with a specific calendar period and fiscal year. The period context references the associated plan context, thus forming a chain. Each plan context has a corresponding set of period contexts.
<b>plan</b>	In PeopleSoft Sales Incentive Management, a collection of allocation rules, variables, steps, sections, and incentive rules that instruct the PeopleSoft Enterprise Incentive Management engine in how to process transactions.
<b>plan context</b>	In PeopleSoft Enterprise Incentive Management, correlates a participant with the compensation plan and node to which the participant is assigned, enabling the PeopleSoft Enterprise Incentive Management system to find anything that is associated with the node and that is required to perform compensation processing. Each participant, node, and plan combination represents a unique plan context—if three participants are on a compensation structure, each has a different plan context. Configuration plans are identified by plan contexts and are associated with the participants that refer to them.
<b>plan template</b>	In PeopleSoft Enterprise Incentive Management, the base from which a plan is created. A plan template contains common sections and variables that are inherited by all plans that are created from the template. A template may contain steps and sections that are not visible in the plan definition.
<b>planned learning</b>	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's planned learning activities and programs.
<b>planning instance</b>	In PeopleSoft Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.
<b>portal registry</b>	In PeopleSoft applications, the portal registry is a tree-like structure in which content references are organized, classified, and registered. It is a central repository that defines both the structure and content of a portal through a hierarchical, tree-like structure of folders useful for organizing and securing content references.
<b>price list</b>	In PeopleSoft Enterprise Pricer, enables you to select products and conditions for which the price list applies to a transaction. During a transaction, the system either determines the product price based on the predefined search hierarchy for the transaction or uses the product's lowest price on any associated, active price lists. This price is used as the basis for any further discounts and surcharges.
<b>price rule</b>	In PeopleSoft Enterprise Pricer, defines the conditions that must be met for adjustments to be applied to the base price. Multiple rules can apply when conditions of each rule are met.

<b>price rule condition</b>	In PeopleSoft Enterprise Pricer, selects the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields are related to the transaction.
<b>price rule key</b>	In PeopleSoft Enterprise Pricer, defines the fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.
<b>process category</b>	In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.
<b>process group</b>	In PeopleSoft Financials, a group of application processes (performed in a defined order) that users can initiate in real time, directly from a transaction entry page.
<b>process definition</b>	Process definitions define each run request.
<b>process instance</b>	A unique number that identifies each process request. This value is automatically incremented and assigned to each requested process when the process is submitted to run.
<b>process job</b>	You can link process definitions into a job request and process each request serially or in parallel. You can also initiate subsequent processes based on the return code from each prior request.
<b>process request</b>	A single run request, such as a Structured Query Report (SQR), a COBOL or Application Engine program, or a Crystal report that you run through PeopleSoft Process Scheduler.
<b>process run control</b>	A PeopleTools variable used to retain PeopleSoft Process Scheduler values needed at runtime for all requests that reference a run control ID. Do not confuse these with application run controls, which may be defined with the same run control ID, but only contain information specific to a given application process request.
<b>product category</b>	In PeopleSoft Enterprise Incentive Management, indicates an application in the Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.
<b>programs</b>	In PeopleSoft Enterprise Learning Management, a high-level grouping that guides the learner along a specific learning path through sections of catalog items. PeopleSoft Enterprise Learning Systems provides two types of programs—curricula and certifications.
<b>progress log</b>	In PeopleSoft Services Procurement, tracks deliverable-based projects. This is similar to the time sheet in function and process. The service provider contact uses the progress log to record and submit progress on deliverables. The progress can be logged by the activity that is performed, by the percentage of work that is completed, or by the completion of milestone activities that are defined for the project.
<b>project transaction</b>	In PeopleSoft Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.
<b>promotion</b>	In PeopleSoft Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.
<b>publishing</b>	In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.
<b>record group</b>	A set of logically and functionally related control tables and views. Record groups help enable TableSet sharing, which eliminates redundant data entry. Record groups ensure that TableSet sharing is applied consistently across all related tables and views.
<b>record input VAT flag</b>	Abbreviation for <i>record input value-added tax flag</i> . Within PeopleSoft Purchasing, Payables, and General Ledger, this flag indicates that you are recording input VAT

on the transaction. This flag, in conjunction with the record output VAT flag, is used to determine the accounting entries created for a transaction and to determine how a transaction is reported on the VAT return. For all cases within Purchasing and Payables where VAT information is tracked on a transaction, this flag is set to Yes. This flag is not used in PeopleSoft Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Expenses, where it is assumed that you are always recording only input VAT.

<b>record output VAT flag</b>	Abbreviation for <i>record output value-added tax flag</i> . See <i>record input VAT flag</i> .
<b>reference data</b>	In PeopleSoft Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, channels, and so on.
<b>reference object</b>	In PeopleSoft Enterprise Incentive Management, this dimension-type object further defines the business. Reference objects can have their own hierarchy (for example, product tree, customer tree, industry tree, and geography tree).
<b>reference transaction</b>	In commitment control, a reference transaction is a source transaction that is referenced by a higher-level (and usually later) source transaction, in order to automatically reverse all or part of the referenced transaction's budget-checked amount. This avoids duplicate postings during the sequential entry of the transaction at different commitment levels. For example, the amount of an encumbrance transaction (such as a purchase order) will, when checked and recorded against a budget, cause the system to concurrently reference and relieve all or part of the amount of a corresponding pre-encumbrance transaction, such as a purchase requisition.
<b>regional sourcing</b>	In PeopleSoft Purchasing, provides the infrastructure to maintain, display, and select an appropriate vendor and vendor pricing structure that is based on a regional sourcing model where the multiple ship to locations are grouped. Sourcing may occur at a level higher than the ship to location.
<b>relationship object</b>	In PeopleSoft Enterprise Incentive Management, these objects further define a compensation structure to resolve transactions by establishing associations between compensation objects and business objects.
<b>remote data source data</b>	Data that is extracted from a separate database and migrated into the local database.
<b>REN server</b>	Abbreviation for <i>real-time event notification server</i> in PeopleSoft MultiChannel Framework.
<b>requester</b>	In PeopleSoft eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.
<b>role</b>	Describes how people fit into PeopleSoft Workflow. A role is a class of users who perform the same type of work, such as clerks or managers. Your business rules typically specify what user role needs to do an activity.
<b>role user</b>	A PeopleSoft Workflow user. A person's role user ID serves much the same purpose as a user ID does in other parts of the system. PeopleSoft Workflow uses role user IDs to determine how to route worklist items to users (through an email address, for example) and to track the roles that users play in the workflow. Role users do not need PeopleSoft user IDs.
<b>roll up</b>	In a tree, to roll up is to total sums based on the information hierarchy.
<b>run control</b>	A run control is a type of online page that is used to begin a process, such as the batch processing of a payroll run. Run control pages generally start a program that manipulates data.
<b>run control ID</b>	A unique ID to associate each user with his or her own run control table entries.

<b>run-level context</b>	In PeopleSoft Enterprise Incentive Management, associates a particular run (and batch ID) with a period context and plan context. Every plan context that participates in a run has a separate run-level context. Because a run cannot span periods, only one run-level context is associated with each plan context.
<b>search query</b>	You use this set of objects to pass a query string and operators to the search engine. The search index returns a set of matching results with keys to the source documents.
<b>section</b>	In PeopleSoft Enterprise Incentive Management, a collection of incentive rules that operate on transactions of a specific type. Sections enable plans to be segmented to process logical events in different sections.
<b>security event</b>	In commitment control, security events trigger security authorization checking, such as budget entries, transfers, and adjustments; exception overrides and notifications; and inquiries.
<b>serial genealogy</b>	In PeopleSoft Manufacturing, the ability to track the composition of a specific, serial-controlled item.
<b>serial in production</b>	In PeopleSoft Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.
<b>session</b>	In PeopleSoft Enterprise Learning Management, a single meeting day of an activity (that is, the period of time between start and finish times within a day). The session stores the specific date, location, meeting time, and instructor. Sessions are used for scheduled training.
<b>session template</b>	In PeopleSoft Enterprise Learning Management, enables you to set up common activity characteristics that may be reused while scheduling a PeopleSoft Enterprise Learning Management activity—characteristics such as days of the week, start and end times, facility and room assignments, instructors, and equipment. A session pattern template can be attached to an activity that is being scheduled. Attaching a template to an activity causes all of the default template information to populate the activity session pattern.
<b>setup relationship</b>	In PeopleSoft Enterprise Incentive Management, a relationship object type that associates a configuration plan with any structure node.
<b>share driver expression</b>	In PeopleSoft Business Planning, a named planning method similar to a driver expression, but which you can set up globally for shared use within a single planning application or to be shared between multiple planning applications through PeopleSoft Enterprise Warehouse.
<b>single signon</b>	With single signon, users can, after being authenticated by a PeopleSoft application server, access a second PeopleSoft application server without entering a user ID or password.
<b>source transaction</b>	In commitment control, any transaction generated in a PeopleSoft or third-party application that is integrated with commitment control and which can be checked against commitment control budgets. For example, a pre-encumbrance, encumbrance, expenditure, recognized revenue, or collected revenue transaction.
<b>SpeedChart</b>	A user-defined shorthand key that designates several ChartKeys to be used for voucher entry. Percentages can optionally be related to each ChartKey in a SpeedChart definition.
<b>SpeedType</b>	A code representing a combination of ChartField values. SpeedTypes simplify the entry of ChartFields commonly used together.
<b>staging</b>	A method of consolidating selected partner offerings with the offerings from the enterprise's other partners.

<b>statutory account</b>	Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft, this is equivalent to the Alternate Account (ALTACCT) ChartField.
<b>step</b>	In PeopleSoft Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.
<b>storage level</b>	In PeopleSoft Inventory, identifies the level of a material storage location. Material storage locations are made up of a business unit, a storage area, and a storage level. You can set up to four storage levels.
<b>subcustomer qualifier</b>	A value that groups customers into a division for which you can generate detailed history, aging, events, and profiles.
<b>Summary ChartField</b>	You use summary ChartFields to create summary ledgers that roll up detail amounts based on specific detail values or on selected tree nodes. When detail values are summarized using tree nodes, summary ChartFields must be used in the summary ledger data record to accommodate the maximum length of a node name (20 characters).
<b>summary ledger</b>	An accounting feature used primarily in allocations, inquiries, and PS/nVision reporting to store combined account balances from detail ledgers. Summary ledgers increase speed and efficiency of reporting by eliminating the need to summarize detail ledger balances each time a report is requested. Instead, detail balances are summarized in a background process according to user-specified criteria and stored on summary ledgers. The summary ledgers are then accessed directly for reporting.
<b>summary time period</b>	In PeopleSoft Business Planning, any time period (other than a base time period) that is an aggregate of other time periods, including other summary time periods and base time periods, such as quarter and year total.
<b>summary tree</b>	A tree used to roll up accounts for each type of report in summary ledgers. Summary trees enable you to define trees on trees. In a summary tree, the detail values are really nodes on a detail tree or another summary tree (known as the <i>basis</i> tree). A summary tree structure specifies the details on which the summary trees are to be built.
<b>syndicate</b>	To distribute a production version of the enterprise catalog to partners.
<b>system function</b>	In PeopleSoft Receivables, an activity that defines how the system generates accounting entries for the general ledger.
<b>TableSet</b>	A means of sharing similar sets of values in control tables, where the actual data values are different but the structure of the tables is the same.
<b>TableSet sharing</b>	Shared data that is stored in many tables that are based on the same TableSets. Tables that use TableSet sharing contain the SETID field as an additional key or unique identifier.
<b>target currency</b>	The value of the entry currency or currencies converted to a single currency for budget viewing and inquiry purposes.
<b>template</b>	A template is HTML code associated with a web page. It defines the layout of the page and also where to get HTML for each part of the page. In PeopleSoft, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.
<b>territory</b>	In PeopleSoft Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.
<b>TimeSpan</b>	A relative period, such as year-to-date or current period, that can be used in various PeopleSoft General Ledger functions and reports when a rolling time frame, rather

	than a specific date, is required. TimeSpans can also be used with flexible formulas in PeopleSoft Projects.
<b>trace usage</b>	In PeopleSoft Manufacturing, enables the control of which components will be traced during the manufacturing process. Serial- and lot-controlled components can be traced. This is maintained in the Item Master record.
<b>transaction allocation</b>	In PeopleSoft Enterprise Incentive Management, the process of identifying the owner of a transaction. When a raw transaction from a batch is allocated to a plan context, the transaction is duplicated in the PeopleSoft Enterprise Incentive Management transaction tables.
<b>transaction state</b>	In PeopleSoft Enterprise Incentive Management, a value assigned by an incentive rule to a transaction. Transaction states enable sections to process only transactions that are at a specific stage in system processing. After being successfully processed, transactions may be promoted to the next transaction state and “picked up” by a different section for further processing.
<b>Translate table</b>	A system edit table that stores codes and translate values for the miscellaneous fields in the database that do not warrant individual edit tables of their own.
<b>tree</b>	The graphical hierarchy in PeopleSoft systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.
<b>unclaimed transaction</b>	In PeopleSoft Enterprise Incentive Management, a transaction that is not claimed by a node or participant after the allocation process has completed, usually due to missing or incomplete data. Unclaimed transactions may be manually assigned to the appropriate node or participant by a compensation administrator.
<b>universal navigation header</b>	Every PeopleSoft portal includes the universal navigation header, intended to appear at the top of every page as long as the user is signed on to the portal. In addition to providing access to the standard navigation buttons (like Home, Favorites, and signoff) the universal navigation header can also display a welcome message for each user.
<b>user interaction object</b>	In PeopleSoft Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All Sales Incentive Management user interface objects and reports are registered as user interaction objects. User interaction objects can be linked to a compensation structure node through a compensation relationship object (individually or as groups).
<b>variable</b>	In PeopleSoft Sales Incentive Management, the intermediate results of calculations. Variables hold the calculation results and are then inputs to other calculations. Variables can be plan variables that persist beyond the run of an engine or local variables that exist only during the processing of a section.
<b>VAT exception</b>	Abbreviation for <i>value-added tax exception</i> . A temporary or permanent exemption from paying VAT that is granted to an organization. This terms refers to both VAT exoneration and VAT suspension.
<b>VAT exempt</b>	Abbreviation for <i>value-added tax exempt</i> . Describes goods and services that are not subject to VAT. Organizations that supply exempt goods or services are unable to recover the related input VAT. This is also referred to as exempt without recovery.
<b>VAT exoneration</b>	Abbreviation for <i>value-added tax exoneration</i> . An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.
<b>VAT suspension</b>	Abbreviation for <i>value-added tax suspension</i> . An organization that has been granted a temporary exemption from paying VAT.
<b>warehouse</b>	A PeopleSoft data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.

<b>work order</b>	In PeopleSoft Services Procurement, enables an enterprise to create resource-based and deliverable-based transactions that specify the basic terms and conditions for hiring a specific service provider. When a service provider is hired, the service provider logs time or progress against the work order.
<b>worksheet</b>	A way of presenting data through a PeopleSoft Business Analysis Modeler interface that enables users to do in-depth analysis using pivoting tables, charts, notes, and history information.
<b>worklist</b>	The automated to-do list that PeopleSoft Workflow creates. From the worklist, you can directly access the pages you need to perform the next action, and then return to the worklist for another item.
<b>XML schema</b>	An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.
<b>yield by operation</b>	In PeopleSoft Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.
<b>zero-rated VAT</b>	Abbreviation for <i>zero-rated value-added tax</i> . A VAT transaction with a VAT code that has a tax percent of zero. Used to track taxable VAT activity where no actual VAT amount is charged. Organizations that supply zero-rated goods and services can still recover the related input VAT. This is also referred to as exempt with recovery.



# Index

## A

- actions
  - create 103
  - data input 110
  - SOAPAction header 126
  - staging and submission 111
  - template actions 106
  - update 103
  - updateData 103
- ActiveX controls
  - errors 117
  - unsupported events 91
- additional documentation x
- alternate search keys 6
- Altkey property 24
- API, *See* application programming interface
- Application Designer
  - APIs 61
  - building APIs 55
  - C++ templates 63
  - COM 69
  - Component Interface Tester 45
  - creating definitions 5
  - Java templates 56
  - PeopleCode template 93
  - using views 5
  - validating component interfaces 43
- application fundamentals ix
- application programming interfaces 1
  - accessing C++ APIs 62
  - bindings 55
  - building in Java 55
  - C++ 61
  - COM 69
  - component interface API 3, 47, 48
  - dummy rows 46
  - naming rules 29
- architecture 3
- attributes
  - collections 4
  - keys 4
  - methods 5
  - name 4
  - overview 3
  - properties 4

- automatic field defaults 10
- autoregister 69

## B

- backpointers 31
- batch processes 18
- bindings
  - C++ 61
  - COM 70
  - early-binding 71, 82
  - Integration Broker 120
  - Java 55, 56
  - late-binding 71
  - PeopleSoft API 55, 70
  - WSDL 119
- buffers 13
- building a component interface 98

## C

- C++
  - building APIs 61
  - client setup 62
  - configuring compilers 62
  - memory conflicts 53
  - requirements 61
  - runtime code templates 63
  - third-party applications 62
- Cancel method 5, 34
- cell formatting 110
- character fields 11
- check boxes 11
- CheckMenuItem 91
- child records 108
- child scrolls 31
- chunking
  - Excel to Component Interface 104
  - log files 117
- chunking factor 103
- clean-up registry 69
- Clear Template button 107
- client-only PeopleCode 92
- collection methods
  - rules 49
  - table of 37
  - testing 49

- collection name 105
  - collection properties 26
  - collections
    - CreateKeyInfoCollection 16
    - defined 4
    - empty 13
    - FindKeyInfoCollection 16
    - GetKeyInfoCollection 16
    - overview 31
    - restrictions 11
    - search key 15
    - user-defined 4
  - collections icon 7
  - COM (Component Object Model) 37
    - APIs 69
    - ASP sample 84
    - Excel sample 82
    - memory conflicts 53
    - overview 1
    - requirements 70
    - runtime code templates 71
    - SDK Excel sample 82
    - third-party applications 70
    - type libraries 70
  - COM Type Library 69
  - comments, submitting xiii
  - CommitWork 98
  - common elements xiii
  - compiler configuration 62
  - %CompIntfc 98
  - CompIntfcName 18
  - %CompIntfcName 98
  - component buffer 13
  - component interface API 3, 47, 48
  - component interface methods
    - standard methods 41
    - user defined 42
  - component interface security 123
  - Component Interface Tester
    - editing history items 46
    - Enter key values dialog box 45
    - Find Results dialog box 47
    - getting dummy rows 46
    - history items 46
    - ItemByKeys parameters 50
    - ItemByKeys: Enter parameters dialog box 50
    - modes 46
    - procedures 44
    - Test dialog box 48
    - testing collection methods 49
    - testing methods 48
    - testing properties 48
  - component interface view 5, 6
  - component interfaces
    - adding to a menu 12
    - creating 10
    - exposing fields 13
    - setting security 43
    - testing 44, 99
    - validating 43
  - Component Object Model, *See* COM (Component Object Model)
  - component transfers 91
  - component view 5
  - ComponentName 19
  - Connect method 33
  - Connection Information tab in Excel to Component Interface 102
  - connection settings
    - defaults 102
    - Excel to Component Interface 101
    - HTTP port 102
    - login screen 104
    - portal name 103
    - protocol 102
    - web server 102
  - Connector ID field 122
  - contact information xiii
  - CopyRowset 92
  - Count method 37
  - Count property 27
  - Coversheet tab in Excel to Component Interface 101
  - create action 103
  - create keys 4, 10, 13, 14, 103
  - Create method 34
  - create new 46
  - Create Reference dialog box 30
  - CreateKeyInfoCollection 16
  - createSession 32
  - cross-references xii
  - CurrentItem collection method 50
  - CurrentItem method 39
  - CurrentItemNum method 40
  - Customer Connection website x
- D**
- data input actions toolbar 110
  - Data Input sheet

- copying to other worksheets 97
- creating 107
- entering data 109
- limitations 111
- overview 109
- Data Item property 27
- data submission 110
- date fields 11
- dates 110
- datetime fields 11
- debugging 18, 104
- DecimalPosition property 23
- decimals 106
- defaults
  - automatic 10, 99
  - connection settings 102
  - Excel to Component Interface 107, 108
  - fields 108
  - properties 10, 11
- definition name 11
- definitions
  - design-time vs. runtime 4
  - in Application Designer 5
- DeleteItem(index) collection method 49
- DeleteItem(Index) method 38
- deleting
  - child properties 31
  - DeleteItem(index) 49
  - DeleteItem(Index) 38
  - keys 13
  - properties 13, 28
- deleting elements 7
- derived properties 28
- Description property 19
- Deselect Input Cell button 107
- design-time properties 4
- DisableMenuItem 91
- Do Not Include for Submission
  - button 107
- Document Object Model (DOM) 97
- documentation
  - printed x
  - related x
  - updates x
- DoModalPageGroup 91
- DoSave() 91
- DoSaveNow() 91
- drop-down list boxes 11
- dummy rows 46
- dynamic tree controls 91

**E**

- edit boxes 11
- edit history items 46
- Edit Property dialog box 28
- EditHistoryItems 17
- EIPs 124
- email 53
- EnableMenuItem 91
- enabling macros 82
- entering data 109
- errors
  - correcting 111, 112
  - Data Input tab 110
  - diagnosing 117
  - in Excel to Component Interface 112
  - list of messages 117
  - log files 117
  - message log 48
  - message logs 104, 117
  - PSProperties not loaded from file 56
  - setting thresholds 103
  - submitting existing keys 112
  - thresholds 104
  - validation 54
- Excel, *See* Microsoft Excel
- Excel to Component Interface
  - building a component interface 98
  - Connection Information tab 102
  - correcting data 112
  - Coversheet tab 101
  - Data Input sheet 109
  - enabling macros 100
  - entering connection settings 101
  - error messages 117
  - Excel limitations 111
  - Excel restrictions 98
  - logging in 104
  - Login dialog box 104
  - PeopleCode limitations 99
  - performance 99
  - Staging and Submission tab 111
  - Template tab 105
  - templates 105

**F**

- field defaults
  - automatic 10
  - criteria for 11
- FieldName property 21

fields

- automatic default criteria 11
- decimals 106
- defaults 10
- exposing 13
- integers 106
- length 106
- names 7
- standard types 106

find keys 4, 14

Find method 5, 34

FindKeyInfoCollection 16

Format property 22

functions

- ASP funclib 85
- Build API names 62
- createSession 32
- ignored 92
- PeopleCode 42
- PeopleCode limitations 91
- restrictions 99
- session 32
- web library 123

**G**

Gateway ID field 121

gateways

- integration 126
- local 120
- setup 120

get existing 46

get history items 46

get keys 4, 14

Get method 5, 35

getCompIntfc 33

GetDummyRow 46

GetDummyRows 20

GetEffectiveItem method 40

GetEffectiveItem(DateString, SeqNum) collection method 50

GetEffectiveItemNum method 40

GetEffectiveItemNum(DateString, SeqNum) collection method 50

GetHistoryItems 17

GetKeyInfoCollection 16

getOAType() 25

GetPropertyByName method 35

GetPropertyInfoByName(). Enter parameters: dialog box 49

GetPropertyInfoByName method 36

getType() 25

glossary 149

**H**

hidden edit validation error 54

HideMenuItem 91

hierarchy 108

HTTP port 102

HttpListeningConnector 126

HTTPPort 115

**I**

IB\_CHNL 120

icons 6

ig.isc.password 121

ig.isc.serverURL 121

ig.isc.toolsRel 121

ig.isc.userid 121

Include All for Submission button 107

Include for Submission button 107

infinite processing 53

Insert New Child button 107

InsertItem method 38

InsertItem(index) collection method 49

InSync messages 122

integration

- accessing external systems 3
- gateway 121
- gateway user information 126
- SDK example 75

Integration Broker

- creating third-party message nodes 121
- gateway setup 120
- modifying local gateway 120
- setup 119
- verifying message channel 120
- WSDL bindings 120

Integration SDK

- C++ sample 79
- COM ASP sample 84
- COM sample 82
- install location 75
- requirements 75
- samples 75
- test page 76
- using the Java sample 77

interactive mode

- considerations 54
- debugging 18

- testing component interfaces 46
- UNIX servers 18
- InteractiveMode 18
- invalid component interfaces 43
- invisible fields 11
- IsCollection property 21
- iScript 100, 123, 124
- IsModalPageGroup 91
- IsReadOnly property 24
- Item(index) collection method 49
- Item(Index) method 39
- ItemByKey parameters 50
- ItemByKey(key1, key2, ...) collection method 50
- ItemByKey(keys) method 39
- ItemByName method 38
- ItemNum property 27
- ItemSelected 91

**J**

- Java
  - bindings 55
  - building APIs 55
  - class file 57
  - methods 25
  - object adapter 25
  - requirements 56
  - runtime code template 56
- Java Development Kit (JDK) 56, 62, 70
- Java Virtual Machine (JVM) 56, 62, 70
- Jolt failover 58

**K**

- Key property 22
- keys
  - adding 15
  - adding manually 14
  - creating 10
  - defined 4, 13
  - deleting 15
  - Excel to Component Interfaces 106
  - existing 112
  - exposing 13
  - getting existing records 46
  - icon 6
  - in scrolls 31
  - modifying 14
  - removing 15

**L**

- LabelLong property 21
- LabelShort property 21
- language code 103
- language support 103
- languages
  - CopyRowset considerations 92
  - installed languages for Excel to Component Interface 103
  - language codes 103
  - submitting in several languages 103
- Length property 23
- level-zero records 13
- limitations
  - PeopleCode 99
- Listboxitem property 24
- load balancing 58
- local gateway 120
- log files
  - generating 104
  - viewing 117
- logging in 104
- Login dialog box in Excel to Component Interface 104
- long character fields 11

**M**

- macros
  - COM sample 82
  - enabling 100
- mapping related keys 30
- Market property 19
- memory
  - conflicts 53
  - releasing 60
  - requirements 98
- menu PeopleCode 91
- menus 12
- message channel 120
- message logs 48, 104
- message nodes 121
- MessageName 126
- messages
  - enabling SOAPTOCI 120
  - Excel to Component Interfaces 103
  - PSMessages collection 54
  - requesting versions 122
  - session error messages 94
- methods

- collection 37
- custom method restrictions for WSDL 123
- defined 5
- Java 25
- session 32
- setting security access 123
- standard 5, 33
- testing 48
- user-defined 5
- Microsoft
  - Excel to Component Interface requirements 98
  - Office 2000 98
  - Visual Basic 70, 71, 98
  - Visual C++ 80
  - XML Parser 98
- Microsoft Excel
  - COM sample file location 82
  - Excel to Component Interface utility overview 97
  - physical limitations 97
  - using the COM sample 82
- MMA Partners x
- Mobile Agent
  - synchronization 51
- mobile properties
  - derived 28
  - do not send updates 28
  - send updates 28
- modal transfers 91
- modifying keys 14
- multilingual support 103
- multiple instances 53

**N**

- name column 7
- Name property 20
- naming guidelines 4, 11
- New Data Input button 107
- New Template button 106
- node 103
- Node 115
- noninteractive mode 46
- notes xii
- number fields 11

**O**

- OaType property 22

- object adapter 25
- out-of-sync icon 7
- OutSync messages 122

**P**

- page control types 11
- parent scrolls 31
- passwords
  - encryption utility 121
  - Excel to Component Interface 104
  - ig.isc.password 121
  - Integration Broker 120
  - SOAP 125
  - SOAP request header 115
  - WSDL 124
- PeopleBooks
  - ordering x
- PeopleCode
  - client-only limitations 92
  - CopyRowset 92
  - creating user-defined methods 42
  - generating templates 93
  - ignored functions 92
  - limitations 91
  - templates 94
  - trigger order 3
  - valid variables 94
- PeopleCode limitations 98
- PeopleCode, typographical conventions xi
- PeopleSoft application fundamentals ix
- PeopleSoft site name 103
- peoplesoft\_peoplesoft\_i.h 61
- performance 99
- permission lists 44, 123
- pop-up menus 91
- Portal 115
- portal name 103
- Post Results button 111
- PrePopup 91
- prerequisites ix
- printed documentation x
- Prompt property 23
- properties
  - automatic defaulting 10
  - criteria for automatic defaulting 11
  - defined 4, 105
  - deleting empty 13
  - exposing 13
  - limits 97

- ordering 7
- read-only 31
- reference 29
- renaming 28
- standard 4, 15
- testing 48
- user-defined 4, 27
- Properties dialog box
  - Synchronization tab 52
- Properties field 122
- properties that are keys 6
- property count 27
- property icon 7
- PropertyInfoCollection 20
- protocol 102
- PS\_HOME\excel folder 100
- PSMessages 54
- Published EIPs page 124

## R

- radio buttons 11
- read-only properties 7, 31
- record names 7
- record type
  - defined 105
- RecordName property 21
- reference backpointers 31
- reference paths 30
- reference properties 29
- related display 53
- related documentation x
- related keys 30
- Request Message field 122
- Request Message Version field 122
- RequestingNode 126
- requests
  - sending 115
- required fields 106
- Required property 22
- required property icon 7
- requirements 75
- response types 115
- Restore Input Cells button 107
- row inserts 53
- rules 49
- run-time properties 4
- runtime code templates 63
- runtime exceptions 3

## S

- Save method 5, 35
- scope conflicts 53
- scroll areas 4, 31
- scroll levels 98
- SDK 75
- SDK (Software Development Kit), *See*
  - Integration SDK
- SDK Java sample 77
- SDK testing 76
- SDK\_BUS\_EXP 75, 76
- SDK\_BUS\_EXPENSES 76
- SDK\_BUS\_EXPENSES page 76
- SDK\_CI\_SAMPLES 76
- search dialog processing 91
- search keys 14
- security 43, 100, 123
  - accessing the SDK 76
  - user profiles 1
  - WSDL discovery 123
- Select All Input Cells button 107
- Select Input Cell button 107
- Send Master 126
- sequence number 106
- %Session 94
- session functions 32
- session methods 32
- SetPropertyByName method 36
- signed number fields 11
- Simple Object Access Protocol (SOAP) 119
- site name 103
- SOAP
  - log files 117
  - SOAP 1.2 119
- SOAP document 126
- SOAP messages
  - sending to PeopleSoft 125
  - testing with Send Master 126
- SOAP/XML requests 113
- SOAPAction Header 126
- SOAPTOCI message 120
- SOAPTOCI web library
  - generating log files 104
  - security 123
- software requirements 98
- Stage for Submission button 110
- Staging and Submission Actions
  - toolbar 111

Staging and Submission tab 111  
 standard methods 5, 33, 41  
   disabling 41  
   enabling 41  
 standard methods icon 7  
 standard properties 4, 15  
 StartSendMaster.bat 126  
 status 106  
 StopOnFirstError 18  
 submission statuses 112  
 Submit Data button 111  
 submitting data 110, 111  
 suggestions, submitting xiii  
 synchronization 51

**T**

template actions toolbar 106  
 Template tab in Excel to Component  
 Interface 105  
 templates  
   adding child records 108  
   adding defaults 108  
   C++ 63  
   column limitations 109  
   COM 71  
   creating in Excel to Component  
     Interfaces 105  
   entering data 108  
   Java 57  
   PeopleCode 93, 94  
   testing 108  
   Visual Basic template 72  
   wrapping 109  
 terms 149  
 testing 44  
   SDK testing 76  
 third-party  
   message nodes 121  
 third-party applications 62, 70  
 thresholds for errors 103  
 time fields 11  
 Transaction Type field 122  
 TransferPage 91  
 translated worksheets 103  
 tree controls 91  
 trigger order 3  
 troubleshooting  
   hidden edit validation error 54  
   infinite processing loops 53  
   Microsoft Office 2000 98

Microsoft XML Parser 98  
 Visual Basic 6.0 SP5 98  
 Type property 22  
 typographical conventions xi

**U**

UncheckMenuItem 91  
 UNIX servers 54  
 update action 103  
 updateData action 103  
 URL 126  
 user ID 104  
 user profiles 1  
 user-defined methods 5, 7  
   creating 41  
   restrictions 10  
 user-defined properties 4  
   creating 27  
   deleting 28  
   renaming 28  
 username 126

**V**

validating 43  
 variables 94  
 view columns 7  
 view icons 6  
 viewing submission results 111  
 views 5  
 Visual Basic 71  
   component interface code 72  
 Visual Basic 6.0 SP5 98  
 visual cues xii

**W**

walkpaths 30  
 warnings xii, 112  
 web libraries 123  
 web library functions 123, 126  
 web server name 102  
 Web Services Description Language  
 (WSDL) 119  
 WEBLIB\_SOAPTOCI 100  
 WebServerMachineName 115  
 WinMessage 53  
 worksheets  
   translated versions 103  
 WSDL  
   generating by PeopleSoft search 124

- generating for component
  - interfaces 123
- generating using iScript 124
- PeopleSoft search 123
- setting up Integration Broker 119
- WSDL 1.2 119

**X**

- Xlat property 23
- XML Spy 126

**Y**

- Yesno property 23

