



PeopleTools 8.12 PeopleSoft Business Interlink Application Developer Guide

SKU MTBUr8SP1B 1200

PeopleBooks Contributors: Teams from PeopleSoft Product Documentation and Development.

Copyright © 2001 by PeopleSoft, Inc. All rights reserved.

Printed in the United States of America.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. and is protected by copyright laws. 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, Inc.

This documentation is subject to change without notice, and PeopleSoft, Inc. 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, Inc. in writing.

The copyrighted software that accompanies this documentation 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 documentation, including the disclosure thereof.

PeopleSoft, the PeopleSoft logo, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, Vantive, and Vantive Enterprise are registered trademarks, and *PeopleTalk* and "People power the internet." are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners.

Contents

About This PeopleBook

Before You Begin	viii
Related Documentation	viii
Documentation on the Internet	viii
Documentation on CD-ROM	viii
Hardcopy Documentation	ix
Typographical Conventions and Visual Cues	ix
Comments and Suggestions	xi

Chapter 1

Introduction to Business Interlinks

What is the Business Interlink Architecture	1-1
Running a Business Interlink	1-1
Running an Incoming Business Interlink	1-4
Creating a Business Interlink	1-4
Creating an Incoming Business Interlink	1-5
Example 1: Calculating UPS Shipping Costs	1-5
Example 2: Creating a Calculate Cost Business Interlink	1-8
Design-Time Plug-in Example	1-8
Runtime Plug-in Example	1-9
Business Interlink Definition Example	1-10
Calculate Cost PeopleCode Example	1-11
List of Common Business Interlink Terms	1-12

Chapter 2

Designing a Business Interlink Definition

Designing a Business Interlink Definition	2-1
Input Tab: Setting Inputs For Your Business Interlink Definition	2-5
Criteria Tab: Setting Criteria For Your Business Interlink Definition	2-6
Output Tab: Setting Outputs For Your Business Interlink Definition	2-8
Settings Tab: Setting Configuration Parameters For Your Business Interlink Definition	2-10
Business Interlink Tester Page: Testing Your Business Interlink Definition	2-11

Using the Business Interlink Page	2-15
Supply Chain Restrictions On Drag And Drop From Control Tree	2-16

Chapter 3

Generating the PeopleCode Template

Generating Business Interlink PeopleCode template in an Event	3-1
Generating a Business Interlink PeopleCode Template For an Application Engine Program.....	3-7

Chapter 4

Running the Business Interlink Object

Example of a transaction PeopleCode template.....	4-3
Example of a transaction PeopleCode template having been fully coded	4-6
Diagrams of the Inputs and Outputs for the Transactions.....	4-11
Page for the fully coded transaction PeopleCode template	4-11
Example: PeopleCode template for query sale_orders	4-13

Chapter 5

Creating an Incoming Business Interlink

Requirements to determine if Incoming Business Interlink suits your needs.....	5-1
Setting up the Incoming Business Interlink Environment for an IScript Function.....	5-1
Test your creation using the XMLLink tester	5-4
Writing the IScript Function.....	5-5
Listing of the XML Request.....	5-9
Listing of the XML Response	5-10
Test Example: XML Design-Time Plug-in Used with an Incoming Business Interlink.....	5-12

Chapter 6

Using the Business Interlink Object Methods

Deciding Which Methods To Use	6-1
Executing the Business Interlink Object	6-1
Supporting Standard Input/Output: BIDocs Methods.....	6-2
Input Docs	6-2
Output Docs	6-4
Supporting Batch Input and Output	6-6
Supporting Rowsets	6-6
Using the Flat Table Input/Output Methods.....	6-6
Supporting Dynamic Output	6-6
Using the Incoming Business Interlink Methods and Properties	6-7

State of a Business Interlink Object.....	6-8
Declaring a Business Interlink Object	6-8
Scope of a Business Interlink Object	6-8
Business Interlink Object Methods.....	6-9
AddInputRow	6-9
BulkExecute	6-11
Clear	6-16
Execute	6-17
FetchIntoRecord.....	6-19
FetchIntoRowset	6-22
FetchNextRow.....	6-25
GetFieldCount	6-26
GetFieldType.....	6-28
GetFieldValue	6-30
GetInputDocs	6-31
GetOutputDocs.....	6-32
InputRowset	6-33
MoveFirst	6-35
MoveNext.....	6-37
BIDocs Hierarchical Methods	6-38
AddDoc	6-38
AddNextDoc	6-40
AddValue	6-43
Clear	6-45
GetCount	6-47
GetDoc	6-48
GetNextDoc.....	6-51
GetPreviousDoc	6-54
GetStatus	6-56
GetValue	6-57
MoveToDoc	6-59
ResetCursor	6-61
Incoming Business Interlink BIDocs Methods And Properties	6-62
AddAttribute	6-62
AddComment	6-64
AddProcessInstruction	6-65
AddText.....	6-67
AttributeCount.....	6-68
ChildNodeCount	6-69
CreateElement	6-70

GenXMLString	6-72
GetAttributeName	6-72
GetAttributeValue	6-74
GetNode	6-75
NodeName.....	6-76
NodeType.....	6-77
NodeValue.....	6-79
ParseXMLString	6-80
Business Interlink and Incoming Business Interlink Functions.....	6-81
GetBiDoc.....	6-81
GetInterlink	6-82
IScript Functions.....	6-83
GetContentBody.....	6-84
Interlink Property.....	6-84
StopAtError	6-84
Configuration Parameters	6-85
Business Interlink Plug-in Configuration Parameters	6-85
URL Configuration Parameter	6-86
BIDocValidate Configuration Parameter	6-86

Index

ABOUT THIS PEOPLEBOOK

This PeopleBook covers the concepts of Business Interlinks. It discusses how PeopleSoft application developers use Business Interlinks to integrate PeopleSoft applications with external systems.

This PeopleBook is written for PeopleSoft Application Developers who will be integrating external systems with PeopleSoft by writing PeopleCode to use Business Interlinks. To take full advantage of the information covered in this book, we recommend that you have a basic understanding of how to use PeopleSoft applications. You should also be comfortable using Microsoft® Windows.

Business Interlink objects are created in the Application Designer. This PeopleBook assumes that you are familiar with Application Designer, and that you understand the structure and relationship of the PeopleSoft application components developed in Application Designer.



For more information about Application Designer see Application Designer.

The Business Interlink XML Design-Time Plug-in is written in the XML markup programming language. An in-depth knowledge of XML is not needed, but some knowledge of XML syntax could be helpful. This PeopleBook shows and explains the syntax that you need to write a Business Interlink XML Design-Time Plug-in.

Once a PeopleSoft application developers has written a Business Interlink XML Design-Time Plug-in and tested it in the Application Designer, a Business Interlink SDK programmer writes the Business Interlink Runtime Plug-in.



For more information about writing a Business Interlink Runtime Plug-in, see PeopleSoft Business Interlink Runtime Plug-in Programming Guide.

The chapters in this document are listed below.

Introduction to Business Interlinks provides an overview of Business Interlink Runtime Plug-ins.

Designing a Business Interlink Definition shows how to design a Business Interlink.

Generating the PeopleCode Template shows how to generate a Business Interlink PeopleCode template.

Running the Business Interlink Object shows how to run a Business Interlink Object using PeopleCode.

Creating an Incoming Business Interlink shows how to set up the environment for Incoming Business Interlinks.

Using the Business Interlink Object Methods describes the methods that are associated with the Business Interlink Object. It contains information to help you decide which of these methods you should use, and lists and describes each method. It also describe Business Interlink properties.

Before You Begin

To benefit fully from the information covered in this book, you need to have a basic understanding of how to use PeopleSoft applications. We recommend that you complete at least one PeopleSoft introductory training course.

You should be familiar with navigating around the system and adding, updating, and deleting information 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.

Related Documentation

To add to your knowledge of PeopleSoft applications and tools, you may want to refer to the documentation of the specific PeopleSoft applications your company uses. You can access additional documentation for this release from PeopleSoft Customer Connection (www.peoplesoft.com). We post updates and other items on Customer Connection, as well. In addition, documentation for this release is available on CD-ROM and in hard copy.



Important! Before upgrading, it is *imperative* that you check PeopleSoft Customer Connection for updates to the upgrade instructions. We continually post updates as we refine the upgrade process.

Documentation on the Internet

You can order printed, bound versions of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM. You can order additional copies of the PeopleBooks CDs through the Documentation section of the PeopleSoft Customer Connection Web site:
<http://www.peoplesoft.com/>

You'll also find updates to the documentation for this and previous releases on Customer Connection. Through the Documentation section of 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 delivered on your PeopleBooks CD.

Documentation on CD-ROM

Complete documentation for this PeopleTools release is provided in HTML format on the PeopleTools PeopleBooks CD-ROM. The documentation for the PeopleSoft applications you have purchased appears on a separate PeopleBooks CD for the product line.

Hardcopy Documentation

To order printed, bound volumes of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM, visit the PeopleSoft Press Web site from the Documentation section of PeopleSoft Customer Connection. The PeopleSoft Press Web site is a joint venture between PeopleSoft and Consolidated Publications Incorporated (CPI), our book print vendor.

We make printed documentation for each major release available shortly after the software is first shipped. Customers and partners can order printed PeopleSoft documentation using any of the following methods:

Internet

From the main PeopleSoft Internet site, go to the Documentation section of Customer Connection. You can find order information under the Ordering PeopleBooks topic. Use a Customer Connection ID, credit card, or purchase order to place your order.

PeopleSoft Internet site: <http://www.peoplesoft.com/>.

Telephone

Contact Consolidated Publishing Incorporated (CPI) at **800 888 3559**.

Email

Email CPI at callcenter@conpub.com.

Typographical Conventions and Visual Cues

To help you locate and interpret information, we use a number of standard conventions in our online documentation.

Please take a moment to review the following typographical cues:

`monospace font`

Indicates PeopleCode.

Bold

Indicates field names and other page elements, such as buttons and group box labels, when these elements are documented below the page on which they appear. When we refer to these elements elsewhere in the documentation, we set them in Normal style (not in bold).

We also use boldface when we refer to navigational paths, menu names, or process actions (such as **Save** and **Run**).

Italics

Indicates a PeopleSoft or other book-length publication. We also use italics for *emphasis* and to indicate specific field values. When we cite a field value under the page on which it appears, we use this style: ***field value***.

We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number *0*, not the letter *O*.

KEY+KEY Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press W.

Jump Destination Indicates a jump (also called a link, hyperlink, or hypertext link). Click a jump to move to the jump destination or referenced section.

Cross-references The phrase For more information indicates where you can find additional documentation on the topic at hand. We include the navigational path to the referenced topic, separated by colons (:). Capitalized titles in *italics* indicate the title of a PeopleBook; capitalized titles in normal font refer to sections and specific topics within the PeopleBook. Cross-references typically begin with a jump link. Here's an example:

For more information, see [Documentation on CD-ROM](#) in *About These PeopleBooks*: Related Documentation.

• **Topic list** Contains jump links to all the topics in the section. Note that these correspond to the heading levels you'll find in the Contents window.



Name of Page or
Dialog Box

Opens a pop-up window that contains the named page or dialog box. Click the icon to display the image. Some screen shots may also appear inline (directly in the text).



Text in this bar indicates information that you should pay particular attention to as you work with your PeopleSoft system. If the note is preceded by **Important!**, the note is crucial and includes information that concerns what you need to do for the system to function properly.



Text in this bar indicates For more information cross-references to related or additional information.



Text within this bar indicates a crucial configuration consideration. Pay very close attention to these warning messages.

Comments and Suggestions

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

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

Or send comments by email to the authors of the PeopleSoft documentation at:

DOC@PEOPLESOFT.COM

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions. We are always improving our product communications for you.

CHAPTER 1

Introduction to Business Interlinks

Business Interlinks enable you to perform component-based, realtime integration from PeopleSoft to external systems. Business Interlinks do this by creating synchronous transactions that allow PeopleSoft applications to pass data to and receive data from the external system in real time. You can use Business Interlinks to integrate PeopleSoft with external systems, with another PeopleSoft application, and systems on the internet web.

A transaction consists of a named command with optional named and typed inputs and outputs. The associated external system or the Business Interlink Plug-in understands this command.

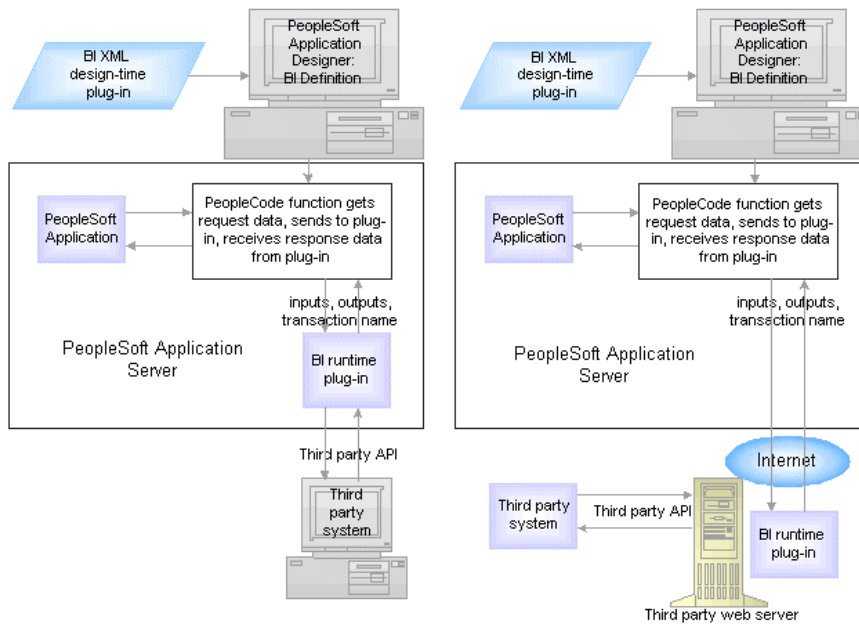
What is the Business Interlink Architecture

The Business Interlink Architecture provides a plug-in framework for PeopleSoft applications to invoke third party APIs over the Internet. Different vendors support different methods for exposing their APIs, including object technologies such as COM, COBRA, EJB; programming language specific interfaces for C or C++; or interfaces based on HTTP and XML. The Business Interlink framework provides a consistent framework for application developers to invoke external applications across this wide variety of technologies.

When a business event triggers the execution of a Business Interlink, the Component Processor synchronously calls the Business Interlink Processor, which in turn invokes the appropriate Business Interlink plug-in. The plug-in provides a wrapper around the third party API and is designed to support any type of interface binding (COM, CORBA, EJB, XML) exposed by the third-party interface. The third-party software could be hosted on the same machine as the PeopleSoft Internet Application Server, or on a separate machine on the other side of the world, invoked over the Internet.

Running a Business Interlink

The following diagram shows the typical Business Interlink architecture, using an XML design-time plug-in and a runtime plug-in.



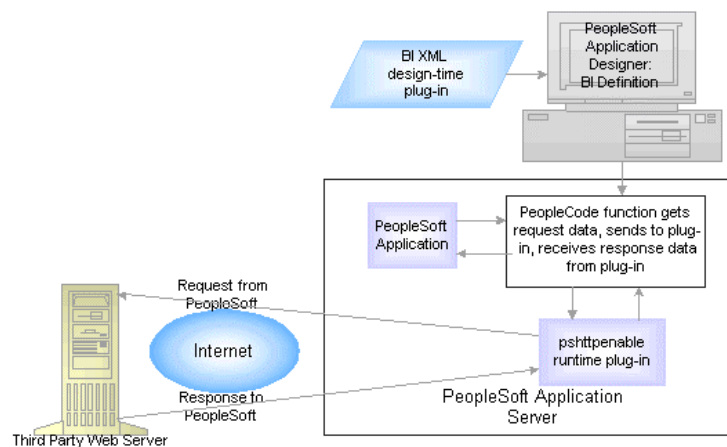
Business Interlink Architecture

When the Business Interlink Object is executed, the following actions take place:

1. A PeopleSoft Application triggers a PeopleCode event. For example, a user might, in a PeopleSoft page labeled “Shipping Time and Cost,” enter input values needed to calculate shipping time, and then press a button labeled “Calculate.”
2. The triggered event (pressing the “Calculate” button) passes the input values to a PeopleCode program, and then executes the PeopleCode program.
3. The PeopleCode program creates an Interlink Object, which contains the transaction name and inputs, and passes the Interlink Object to the Business Interlink runtime plug-in. The runtime plug-in can be located on:
 - A web server
 - The same PeopleSoft Application Server as the PeopleSoft Application.
4. The Business Interlink runtime plug-in provides the implementation of the transactions and/or data operations. It calls the external software system through its API, passing the input values from the Business Interlink Object. This external system can be located on:
 - An external server.
 - A web server.
 - The same PeopleSoft Application Server as the PeopleSoft Application.
5. The third party system performs operations based on those input values, and returns output values through its API to the runtime plug-in. These operations could be, for example, executing functions in the external system, or performing operations on a database in the external system.

- The Business Interlink runtime plug-in assigns output values to the Business Interlink Object (if there are outputs). If there are outputs, the PeopleCode program can assign the output values to PeopleSoft variables. For example, in a PeopleSoft page labeled “Shipping Time and Cost,” the output values could then be displayed upon that page.

There is a specialized type of Business Interlink design-time plug-in, which uses the pshttпенable runtime plug-in. Some external systems receive data via HTTP in the form of an XML or HTML request, and then return data in the form of an XML or HTML response. When your external system uses data in this manner, you can use the pshttпенable runtime plug-in. This means that a Business Interlink runtime plug-in need not be created for your external system. The following diagram shows the architecture for Business Interlinks using the pshttпенable runtime plug-in.



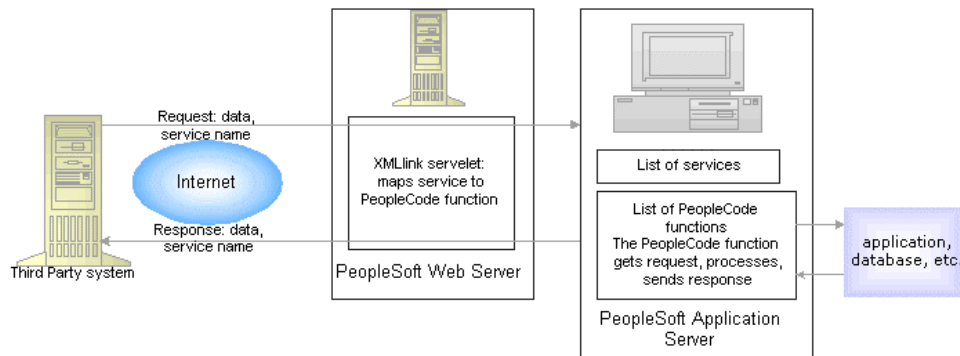
Business Interlink Architecture

When the Business Interlink Object is executed for a Business Interlink using a pshttпенable runtime plug-in, the following actions take place:

- A PeopleSoft Application with input values triggers a PeopleCode event.
- The triggered event passes the input values to a PeopleCode program, and then executes the PeopleCode program.
- The PeopleCode program creates an Interlink Object, which contains the input data and the transaction name, and passes the Interlink Object to the Business Interlink runtime plug-in. The runtime plug-in is the pshttпенable plug-in, located on the PeopleSoft application server.
- The pshttпенable runtime plug-in sends a request over the Internet to the third party web server, passing it input values from the Business Interlink Object.
- The third party web server performs operations based on those inputs, and returns a response over the Internet, passing back output values to the pshttпенable runtime plug-in.
- The Business Interlink runtime plug-in assigns output values to the Business Interlink Object (if there are outputs). If there are outputs, their values can be fetched and assigned to PeopleSoft variables.

Running an Incoming Business Interlink

An Incoming Business Interlink allows PeopleSoft to receive an XML request via HTTP from a third party system, and send an XML response to the third party system. The following diagram shows the architecture for Business Interlinks using the pshttpenable runtime plug-in.



Business Interlink Architecture

When the third party system sends a request over the Internet to PeopleSoft, the following actions take place:

1. A third party system sends an XML request over the Internet to a PeopleSoft web server. This request also contains a service name.
2. The XMLLink servlet on the PeopleSoft web server maps the service name to a PeopleCode function, and executes the function.
3. The PeopleCode program parses the XML request.
4. The PeopleCode program can execute an application, access a database, or other task, depending upon the content of the request. For example, it could search an employee database to find employees who have a certain skill.
5. The PeopleCode program creates an XML response.
6. The PeopleCode program sends the XML response to the third party system.

Creating a Business Interlink

To allow users to create and run Business Interlinks, developers must perform the following tasks. This manual describes the tasks that are in **bold**.

1. A developer designs the shape of a transaction(s) (inputs, outputs, name). For example, the action could be telling UPS to calculate the cost of shipping.
2. A developer writes a Business Interlink design-time plug-in that implements the shape of the transaction(s). This plug-in is written in the XML markup language.

3. A developer writes a Business Interlink runtime plug-in to implement the transaction: passing the inputs to an external system and receiving the outputs from the external system.
4. The design-time plug-in and the runtime plug-in are deployed for use by PeopleSoft applications.
5. A PeopleSoft Application Developer creates a Business Interlink Definition, which creates a specific shape for a particular PeopleSoft application. For example, the application might be created, or modified, to be able to connect to UPS and calculate shipping costs.
6. A PeopleSoft Application Developer writes a PeopleCode program to call the Business Interlink object that is created for the Business Interlink definition.
7. A user can now use the PeopleSoft application to run the Business Interlink. For example, the user can now connect to UPS and calculate shipping costs.

Creating an Incoming Business Interlink

To allow PeopleSoft to use Incoming Business Interlinks, developers must perform the following tasks.

1. A PeopleSoft developer creates an IScript record.
2. The developer writes the PeopleCode function that parses the XML request to extract inputs, and after calling PeopleSoft applications with those inputs, builds an XML response. The XML request and response structure are provided by the third party.
3. The developer registers the function on the PeopleSoft web server by a service name.

Example 1: Calculating UPS Shipping Costs

The Shipping Time & Cost example shows how Business Interlinks can be used. Shipping Time & Cost use the Business Interlink plug-ins that were written for UPS.

http://dgiannmon042400/clientservlet.run - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://dgiannmon042400/clientservlet.run Go

Exit Help

Shipping Time & Cost
United Parcel Service

Ground Time-in-Transit

*Origin: 

*Destination:

QuickCost

*Country Origin: *Zip:

*Country Destination: *Zip:

*Drop off/Pickup Type: *Weight:

*Packaging: lbs.

Service Type	Guaranteed By	Commercial Rate

Local intranet

Shipping Time & Cost Page

In the Shipping and Time page, you can trigger two different Business Interlinks that call out to the UPS shipping and tracking Web site to calculate:

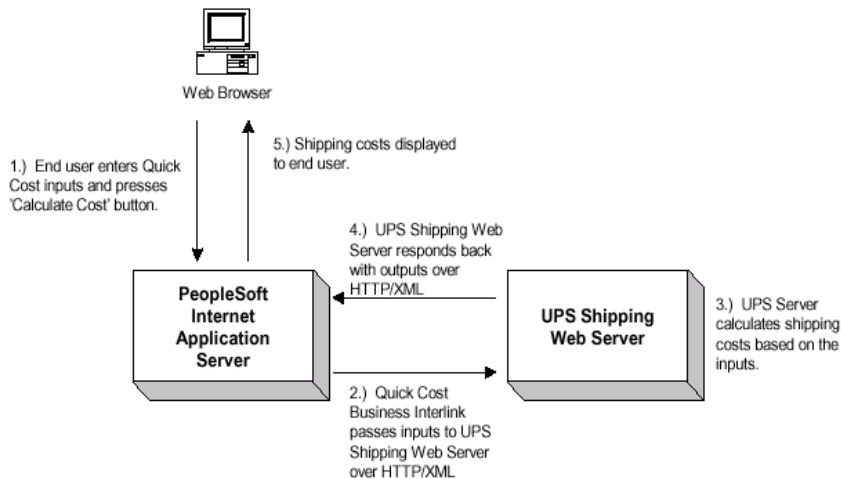
- Ground Time-in-Transit to determine how long it will take to deliver the package.
- Quick Cost to calculate how much it will cost to ship the package based on the shipping service type.

For Quick Cost, you first enter Origin Country and Postal Code, Destination Country, Postal Code, Packaging information, and then you press the Calculate button. At this point, the Business Interlink is invoked on the Application Server, and it issues an HTTP request out to the UPS Web site, which calculates the shipping costs and returns the information back to the PeopleSoft application. Here are the results of a Quick Cost calculation.

Service Type	Guaranteed By	Commercial Rate
UPS Next Day Air Early A.M.	8:30 A.M. - Next Day	\$ 70.25
UPS Next Day Air	10:30 A.M. - Next Day	\$ 45.25
UPS Next Day Air Saver	3:00 P.M. - Next Day	\$ 39.75
UPS 2nd Day Air A.M.	12:00 P.M. - 2 Days	\$ 27.30
UPS 2nd Day Air	End of Day - 2 Days	\$ 24.30
UPS 3 Day Select	End of Day - 3 Days	\$ 16.70
UPS Ground		\$ 7.24

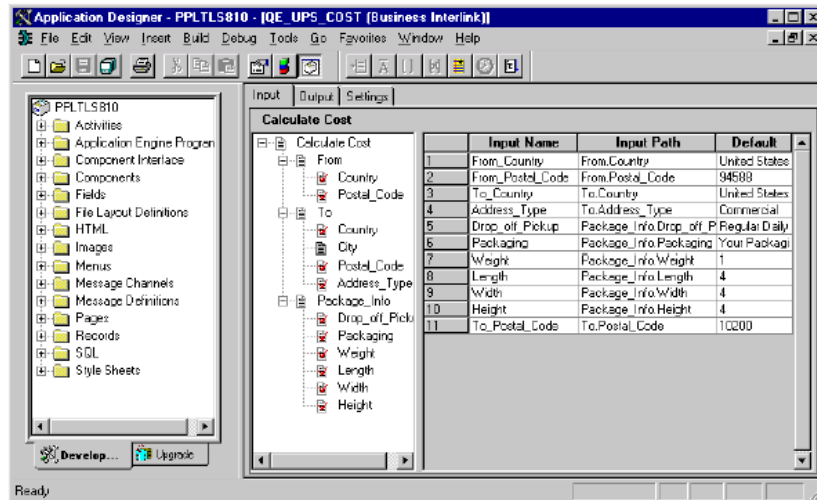
Shipping Time & Cost Page: Calculate Cost Business Interlink

This diagram explains the processing flow of this example in more detail.

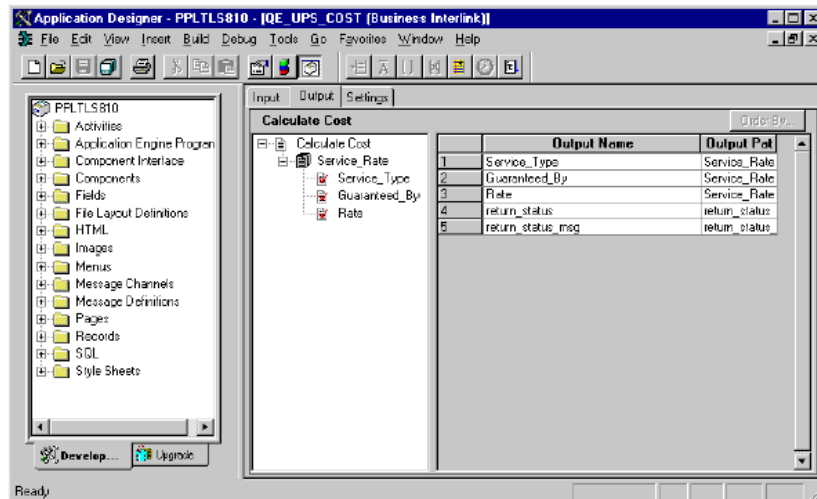


Calculate Cost Business Interlink Processing Flow

In the PeopleTools Application Designer, you can view the Business Interlink Definition for the Quick Cost application. A Business Interlink consists of a set of inputs and outputs. The first screen shows the inputs, and the second shows the outputs.



Calculate Cost Business Interlink Inputs



Calculate Cost Business Interlink Outputs

Example 2: Creating a Calculate Cost Business Interlink

This section contains code segments that overview a Business Interlink that calculates shipping costs.

Design-Time Plug-in Example

The design-time plug-in is written in XML, and defines the shape of the Business Interlink. The shape is the inputs and outputs for the transaction. Using XML for the design-time plug-in easily allows you to easily organize the inputs and outputs into a hierarchical structure.

Following is a simplified XML example for a transaction named Calculate Cost.

```

<transaction name="Calculate Cost">

  <input_list>

    <input name="From">

    <input name="To">

    <input name="Package_Info">

  </input_list>

  <output_list>

    <output name="Service_Rate">

  </output_list>

</transaction>

```

Runtime Plug-in Example

The runtime plug-in can be written in C++, Visual Basic, or Java. It uses the Business Interlink Object as its input and output. the Business Interlink Object containing the inputs, outputs, and Business Interlink methods. Since the runtime plug-in encapsulates the external system, PeopleSoft applications can treat all Business Interlinks the same: as a PeopleSoft object.

The runtime plug-in performs the following tasks:

- Connects to the external system.
- Passes inputs to the external system.
- Receives outputs from the external system.

A runtime plug-in must be written specifically for each external system's interface architecture.

Following is a simplified runtime plug-in pseudo-code example for a transaction named Calculate Cost. This transaction gets inputs from the Business Interlink Object IntObj, passes them to a function provided by the third party that calculates the cost, or ServiceRateValue, and then receives the cost and passes it to the Business Interlink Object.

```

ExecuteTransaction(InterfaceObject IntObj)
{
  docsOut = IntObj->GetOutputDocs();
  docsIn = IntObj->GetInputDocs();

  if(IntObj->GetObjName() == "Calculate Cost") {

```

```
FromValue = docsIn.GetValue("From");

ToValue = docsIn.GetValue("To");

PackageInfoValue = docsIn.GetValue("Package_Info");

ExternalCallToCalcCost(FromValue, ToValue, PackageInfoValue,

    ServiceRateValue)

docsOut.AddValue("Service_Rate", ServiceRateValue)

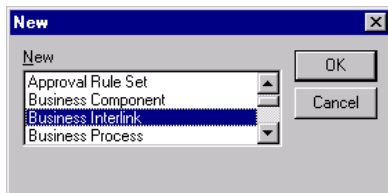
}

}
```

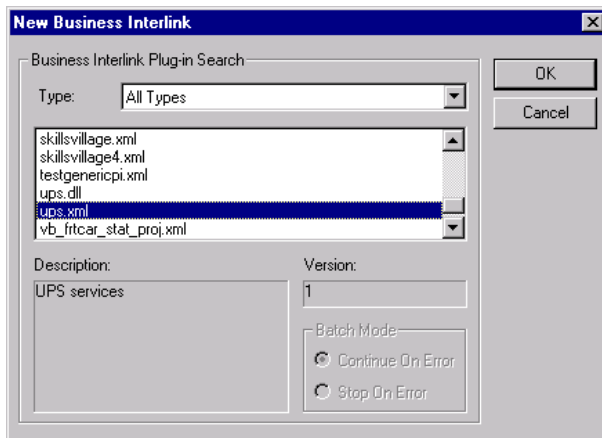
Business Interlink Definition Example

The Business Interlink Definition is created using PeopleSoft Application Designer, and it contains the specific shape that you want to use for your Business Interlink. Within PeopleSoft Application Designer, you select the XML design-time plug-in for your external system, then you select the transaction, and then you select the input and output parameters that you want to use with that transaction (required parameters are selected automatically).

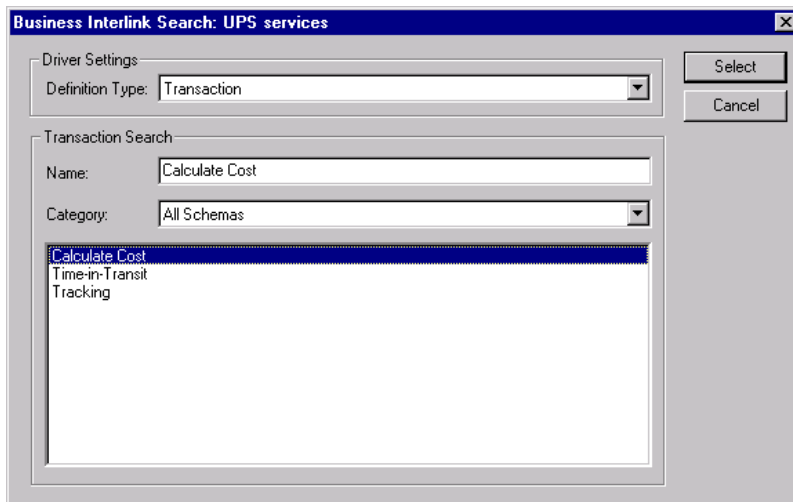
Following is a sample of creating a Business Interlink Definition.



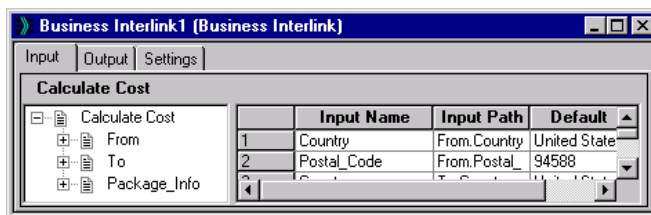
New Popup dialog box, Selecting Business Interlink



New Business Interlink dialog box



Business Interlink Search dialog box



Business Interlink Search dialog box

Calculate Cost PeopleCode Example

Following is a simplified PeopleCode example for a transaction named Calculate Cost. This PeopleCode gets the Business Interlink Definition, BI_COST, and creates a Business Interlink Object, BI_COST_OBJ. The PeopleCode program adds input values to the Interlink Object. The

PeopleCode program executes the Interlink Object, which sends inputs to the runtime plug-in, executes the runtime plug-in, and receives outputs from the runtime plug-in. Then the PeopleCode program gets the output values from the Interlink Object.

```

Local Interlink &BI_COST_OBJ;

Local BIDocs &inDoc;

Local BIDocs &outDoc;

Local boolean &RSLT;

Local float &RATE;

&BI_COST_OBJ = GetInterlink(INTERLINK.BI_COST);

/* ==> Add inputs */

&InputDocs = &BI_COST_OBJ.GetInputDocs("");

&ret = &InputDocs.AddValue("From", "San Francisco");

&ret = &InputDocs.AddValue("To", "New York");

&ret = &InputDocs.AddValue("Package_Info", "Standard");

/* ==> Execute the Interlink */

&EXECSRSLT = &BI_COST_OBJ.Execute();

If ( &EXECSRSLT <> 1 ) Then

    /* The instance failed to execute */

Else

    &outDoc = &BI_COST_OBJ.GetOutputDocs("");

    &ret = &outDoc.GetValue("Service_Rate", &RATE);

End-If; /* If NOT &RSLT ... */

```

List of Common Business Interlink Terms

Application Designer: The integrated development environment used to develop PeopleSoft applications.

Basic Type: A data type such as an integer or string.

Business Interlink Definition: A definition encapsulating an external Transaction or Query and providing a set of generically typed input/outputs that can be assigned to PeopleCode variable or

Record Fields at runtime. A Business Interlink Definition is added to the Application Designer's objects at the same level as Fields, Records, Pages, etc.

XML Design-Time Plug-in: An XML file that, when coded for an external system, encapsulate that external system and provide a catalog of Transactions, Classes and Criteria specific and meaningful to that external system. This functionality can also be written using a set of C++, Visual Basic, or other high-level language methods.

Business Interlink Framework: The framework for integrating any external system with PeopleTools application objects. It is composed of the following components: 1) An External System, 2) Generic definitions for a Transaction/Query command interfaces, 4) Business Interlink Definitions, 4) Business Interlink Plug-in.

Business Interlink Object: an instantiation based on a Business Interlink Definition. Actual data can be added to the inputs of the Business Interlink Objects once the appropriate bindings are provided. The Business Interlink Object can be executed to perform the external service. Once a Business Interlink Object is executed, the user of that object can retrieve the outputs of the external service. The Business Interlink Objects use buffers to receive input and send output. When a Business Interlink Object is executed, the transaction/query/class associated to the Business Interlink Object will be executed once per each row of the input buffers corresponding to the input Records. If there is only one row, after appropriate substitution by the driver, it is executed only once.

Runtime Plug-in: A set of C++, Visual Basic, or other high-level language methods that, when coded for an external system, encapsulate that external system and provide the execution methods to match the Business Interlink Design-Time Plug-in.

Catalog: the list of transactions, classes, and queries used to interface to the external system. Integration users are presented with this list when they pick the type of Business Interlink Plug-in they are going to use. There are four types of catalogs:

- Transaction catalog: lists transactions used to interface to the external system. See Transaction.
- Class catalog: lists classes used to interface to an external system. A class contains data members of basic types and/or objects that are typed after another class. A Class can also contain lists of basic types or objects.
- Operator catalog: lists query operators used to query the external system. These query operators are used to query the classes in the class catalog.
- Configuration parameter catalog: Used to configure an external system with PeopleSoft. For example, it might set up configuration and communication parameters for an external server.

External System: Any system that is not directly compiled with the PeopleTools servers.

PeopleCode: PeopleSoft's proprietary language; it is executed by the PeopleSoft Application Processor. PeopleCode generates results based upon specific actions, based upon existing data or the actions of a user. Business Interlink Objects are executed by calling the execute() method from PeopleCode. This makes external services available to all PeopleSoft applications wherever PeopleCode can be executed.

PeopleCode Event: An action that an end-user takes upon an object, usually a Record Field, that is referenced within a PeopleSoft page.

Query: a set of data members that are selected from a Class catalog (provided by the Business Interlink Plug-in) as well as a generic form of Criteria. The criteria are composed of <left-hand-side> <Relational Operator> <right-hand-side> statements that can be concatenated using a set of logical operators. All operators and class catalogs are dynamically provided through the Business Interlink Plug-in.

Transaction: a named command with optional named and typed inputs and outputs. The associated external system or the Business Interlink Plug-in understands this command. The types of inputs and outputs are based on a set of generic types.

Shape: For a transaction, the set of inputs and outputs for that transaction. For a class, the data members of that class.

CHAPTER 2

Designing a Business Interlink Definition

This section shows how to design a Business Interlink, using a freight carrier plug-in as an example. The example uses a freight carrier Transaction, along with some information about Supply Chain Queries.

This section also shows how to set the inputs, outputs, and configuration parameters for a Business Interlink Definition, and how to test a Business Interlink Definition.

Designing a Business Interlink Definition

You will use the Application Designer to design the Business Interlink Definition. You need to know how to enter the PeopleSoft Application Designer to use these instructions.

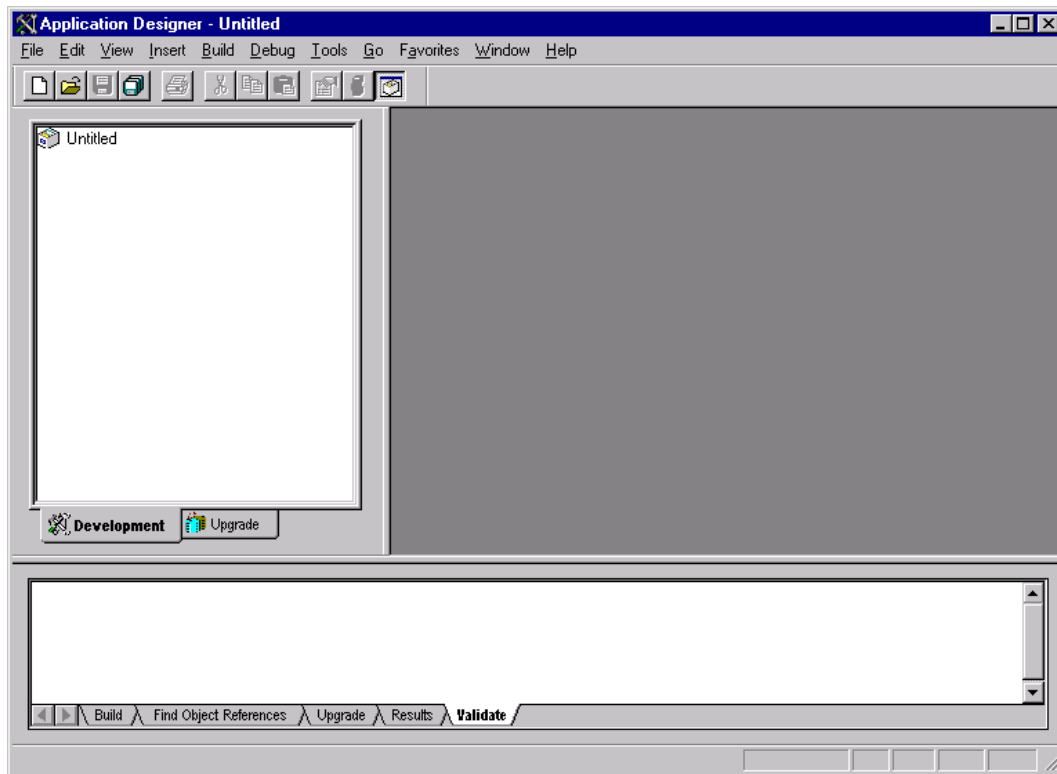


For more information about the Application Designer, see Application Designer.

To design a Business Interlink Definition

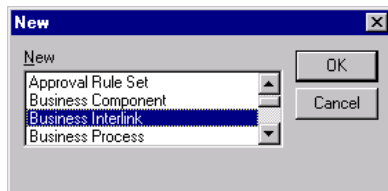
1. Start the PeopleSoft Application Designer.

You will see a screen similar to the following.



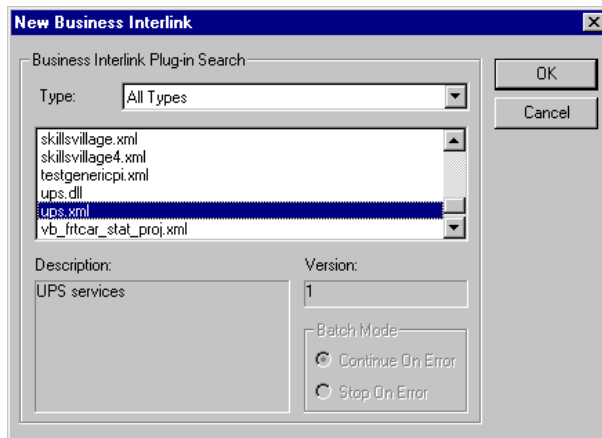
PeopleSoft Application Designer

2. From the File menu, select **New**. Then from the New popup dialog box, select **Business Interlink** and click OK.



New Popup dialog box, Selecting Business Interlink

3. From the New Business Interlink dialog box, select the name of the desired Business Interlink plug-in and click OK. (**ups.xml** is the name for the design-time plug-in used in this example.)



New Business Interlink dialog box

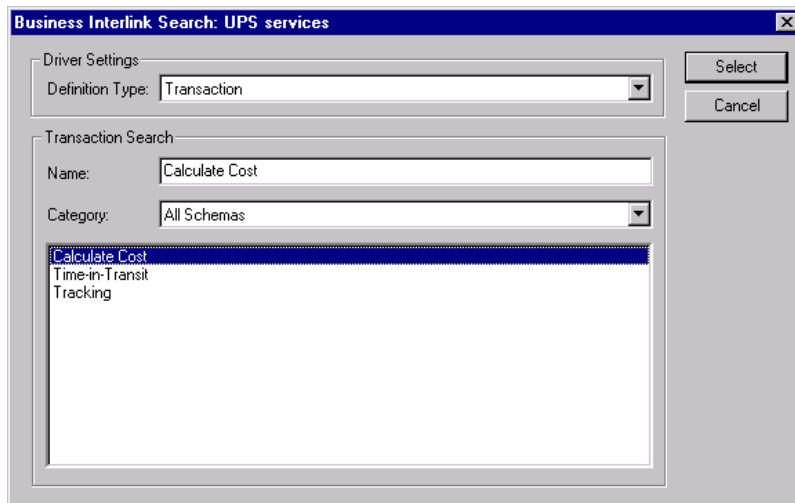
4. From the Business Interlink Search dialog box, select the transaction or query from which you want to build the Business Interlink Definition.

For queries, select **Query** from the Definition Type drop-down list. Then either type the name of the class into the Name field and press Select, or fill the Class Search list box with class names and double-click on the desired name.

For transactions, select **Transaction** from the Definition Type drop-down list. Then either type the name of the transaction into the Name field and press Select, or fill the Transaction Search list box with transaction names and then double-click on the desired name.

To fill the Transaction Search list box or Class Search list box:

- click Select to fill it with all the transaction or class names, or
- select a category from the Category drop-down list to list only the transaction or class names in that category, or
- type the first part of the transaction or class name into the Name field and press Enter to get a list of transaction or class names that start with the typed text.



Business Interlink Search dialog box

5. Set the parameters for this Business Interlink Definition with the Input, Output, or Criteria tabs.
 - For transactions, use the Input and Output tabs to set the input and output parameters for this Business Interlink Definition.
 - For queries, use the Criteria and Output tabs to set the criteria and output parameters for this Business Interlink Definition.



For more information on the Input tab, see [Input Tab: Setting Inputs For Your Business Interlink Definition](#).



For more information on the Output tab, see [Output Tab: Setting Outputs For Your Business Interlink Definition](#).




For more information on the Criteria tab, see [Criteria Tab: Setting Criteria For Your Business Interlink Definition](#).

6. If needed, set the Configuration Parameters with the Settings tab.



For more information on the Settings tab, see [Settings Tab: Setting Configuration Parameters For Your Business Interlink Definition](#).

7. After you have set the Inputs, Criteria, Outputs, and Settings as you want them, select **File: Save** to save your Business Interlink Definition.

8.  your Business Interlink Definition, either select **View:Test** or click the Test button

Next, you will use PeopleCode to instantiate your Business Interlink Definition, create a Business Interlink Object, and run it.

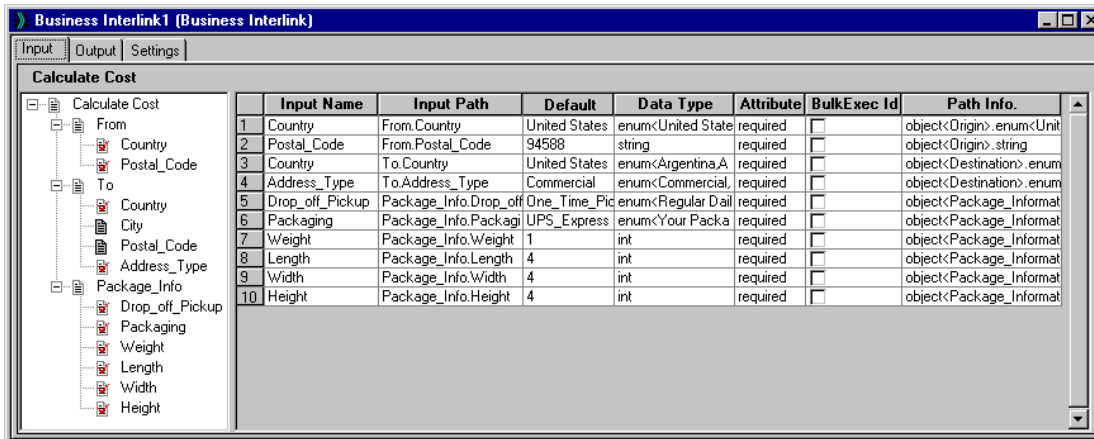


For more information on using PeopleCode this way, see Generating the PeopleCode Template and Running the Business Interlink Object.

Input Tab: Setting Inputs For Your Business Interlink Definition

For transactions, you click the Input tab, and then set the Inputs for your Business Interlink Definition.

In this case, this is the page for Calculate Cost(Domestic). Note that default values are in the Default column; these values were created in the design-time plug-in file, psfedex.xml.



	Input Name	Input Path	Default	Data Type	Attribute	BulkExec Id	Path Info.
1	Country	From.Country	United States	enum<United State	required		object<Origin>.enum<Unit
2	Postal_Code	From.Postal_Code	94588	string	required		object<Origin>.string
3	Country	To.Country	United States	enum<Argentina,A	required		object<Destination>.enum
4	Address_Type	To.Address_Type	Commercial	enum<Commercial	required		object<Destination>.enum
5	Drop_off_Pickup	Package_Info.Drop_off	One_Time_Pic	enum<Regular Dail	required		object<Package_Informat
6	Packaging	Package_Info.Packagi	UPS_Express	enum<Your Packa	required		object<Package_Informat
7	Weight	Package_Info.Weight	1	int	required		object<Package_Informat
8	Length	Package_Info.Length	4	int	required		object<Package_Informat
9	Width	Package_Info.Width	4	int	required		object<Package_Informat
10	Height	Package_Info.Height	4	int	required		object<Package_Informat

Business Interlink Definition: Input Tab

- To set the value for an input (or override the default value for that input), type a value in the Default column. The Default column lists the default value for every input, if one exists.
- You cannot delete Required inputs from the grid. By default, all required inputs are placed in the grid, which selects them for your Business Interlink Definition.
- To select an optional input for your Business Interlink Definition, drag it from the tree control and drop it on the grid. You cannot select an input object (shown by a + or – to the left of its name).
- The Input Name column contains the display name of the input. This is the name that will be used for the input within PeopleCode. You can select and edit the input name in the Input Name column, and the edited name will be used for the input within PeopleCode.

- The BulkExec ID column is used when you use the BulkExecute PeopleCode method instead of the Execute PeopleCode method to execute the Business Interlink. Check the checkboxes that are key fields; these fields are the input record fields that are duplicated in the output record.



For more information about BulkExecute, see BulkExecute.



For more information about the PeopleCode for this example, see the PeopleCode example in Running the Business Interlink Object.

Criteria Tab: Setting Criteria For Your Business Interlink Definition

For queries, you click the Criteria tab, and then set the Criteria for your Business Interlink Definition.

In each grid row, you create query statements of the form:

<Data Member> <Operator> <Input>

Input serves as a placeholder; you later assign a value to it.

The page below shows the Business Interlink Criteria for a Supply Chain query, followed by the syntax for that query.

	Data Member	Operator	Input	Default	Logical	Data Type
1	display_name	<	IN1_display_name		and	string
2	~order_number	<	IN2_order_number		and	string
3	order_date	<	IN3_order_date		and	datetime
4	order_date	>=	IN4_order_date			datetime

Business Interlink Definition: Criteria Tab

```
Select Sales_Order
where display_name = IN1_display_name
AND NOT order_number < IN2_order_number
OR (order_date < IN3_order_date AND order_date >= IN4_order_date)
```


The Inputs (IN1_display_name, IN2_order_number, IN3_order_date, and IN4_order_date) are the right-hand-side of the query statements. You will assign values to the Inputs when you fill in the PeopleCode template for the Business Interlink Object created by generating this Business Interlink Definition.




This is a Supply Chain query. Queries for other plug-ins may use different syntax.


- To select a Data Member, drag it from the tree control and drop it on the grid. You can select data members of basic type for a query; you cannot select a list, an object, an object list, or any data member from an object list (they will be blanked out).
- To select a relational operator, double-click on the Operator field to open a list of operators, and select the operator that you want to use.
- To select a logical operator, double-click on the Logical field to open a list of Logical operators, and select the Logical operator that you want to use. The logical operator acts upon the grid row containing the logical operator, and the grid row immediately below it.




For more information on filling the PeopleCode template, see Running the Business Interlink Object.

Three of the toolbar buttons are used with criteria. These functions can also be accessed by right-clicking on a grid row.

Negate or NOT : Select a grid row and click this button to perform a NOT operation on the query statement in that row. You can also right-click on the row and select Negate from the pop-up menu. When a row is negated, it will have a "~" inserted before its name in the Data Member column, as is shown with order_number in the grid below.

Group Rows or Parenthesis : Select one or more rows and click this button to put parenthesis around these rows. You can also select one or more rows, right-click on the selected rows, and select Group from the pop-up menu. When rows are grouped, they will show a blue left-parenthesis on the left side of the row. The two order_date statements in the grid are grouped this way.

Ungroup : Select grouped rows in the grid and click this button to ungroup them. You can also select one or more rows that are grouped, right-click on the selected rows, and select Group from the pop-up menu.



Criteria Toolbar Buttons

Output Tab: Setting Outputs For Your Business Interlink Definition

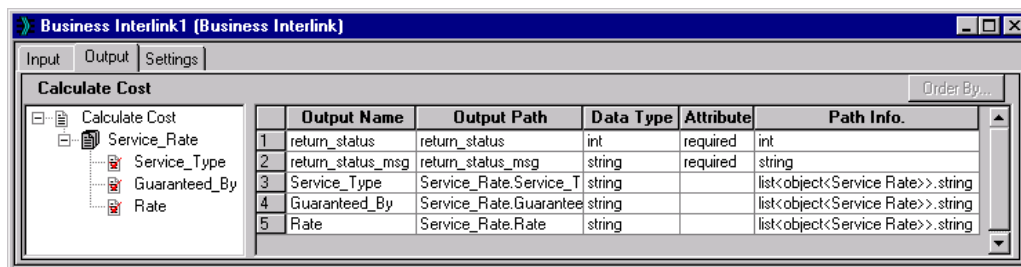
For transactions and queries, click the Output tab, and then set the Outputs for your Business Interlink Definition.

To select an output for your Business Interlink Definition, drag it from the tree control and drop it on the grid.

The Output Name column contains the display name of the output. This is the name that will be used for the output within PeopleCode. You can select and edit the output name in the Output Name column, and the edited name will be used for the output within PeopleCode. You will receive values for the Outputs when you fill in the PeopleCode template for the Business Interlink Object created by generating this Business Interlink Definition.



For more information on filling the PeopleCode template, see [Running the Business Interlink Object](#).



Business Interlink Definition: Output Tab

For query outputs, you can specify the order of the query outputs by clicking on the **Order By** button, which will bring up the Order By Dialog box. The class for this query will be listed in the tree control.

- To select an output from that class for your Business Interlink Definition, drag it from the tree control and drop it on the grid. The order of the query outputs will match the order of the data members in the grid.
- To move the outputs on the grid, highlight the Output Path and click the Up or Down button to move it.

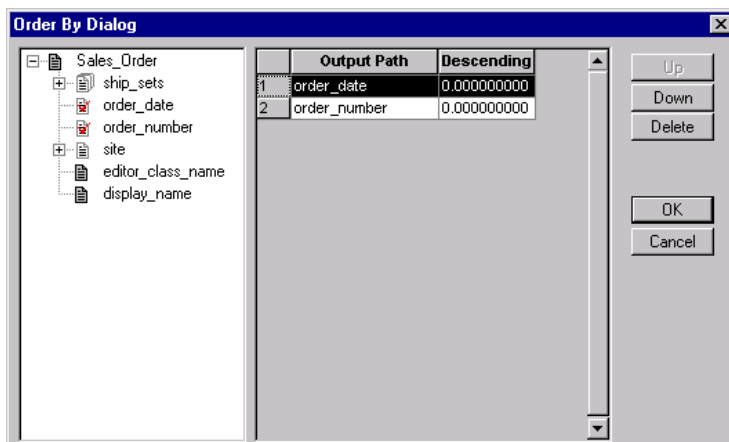
- To determine ascending or descending order, check the checkbox in the Descending column to be in descending order, and leave the checkbox blank for ascending order.

Applying the Popup dialog box below to the sales order done in the previous step (for queries), and then set the Criteria for your Business Interlink Definition), we have the following query:

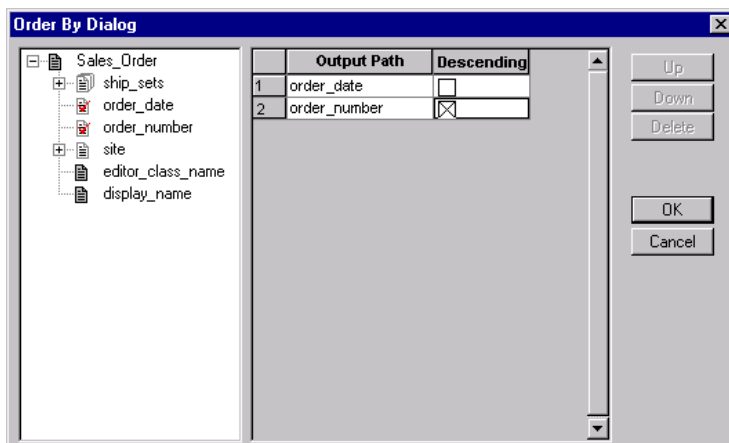


For more information about setting criteria, see Criteria Tab: Setting Criteria For Your Business Interlink Definition.

```
Select Sales_Order where display_name = IN1_display_name
and order_number < IN2_order_number
and (order_date >= IN3_order_date and order_date < IN4_order_date)
order by order_date, order_number-
```



Order By Popup dialog box for Query Outputs



Order By Popup dialog box for Query Outputs (old screen)

Settings Tab: Setting Configuration Parameters For Your Business Interlink Definition

To set the Configuration Parameters, click the Settings tab.

If you are going to test your Business Interlink Definition, you may need to set your Configuration Parameters. For example, you may need to set them to values that connect with a running server.

- To set the Configuration Parameters defaults for this Business Interlink Plug-in, enter the values into the Defaults column.
- To set the defaults of the Configuration Parameters for this driver to the values you have entered in the Defaults column, enter the values and then press the Set Default button to save the values in the driver database. The next person to create a Business Interlink Definition for this Business Interlink Plug-in will have the Configuration Parameter defaults that you entered here. These default values are also used when you instantiate the Business Interlink Definition, creating a Business Interlink Object, and then run that Business Interlink Object using PeopleCode. You can override these default values in PeopleCode.

The configuration parameters for this plug-in are:

URL

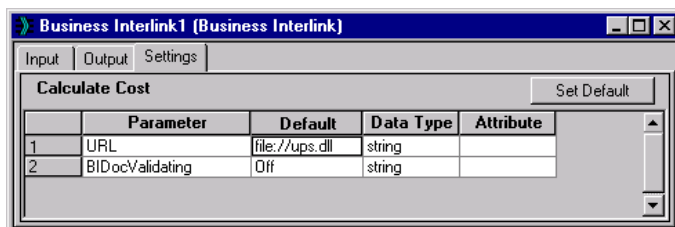
Set to the location of the Business Interlink runtime plug-in.

BIDocValidate

When set to ON, a BIDocs object is checked to see if it exists before adding/getting values from it. For example, if the BIDoc object "Rate" does not exist, and BIDocValidate is set to ON, then the following line of code will cause an error:

```
&ret = &biDocs.GetValue("Rate", &RATE);
```

For more information about the BIDocs methods, see PeopleSoft Business Interlink User Guide, "Using the Business Interlink Object Methods".



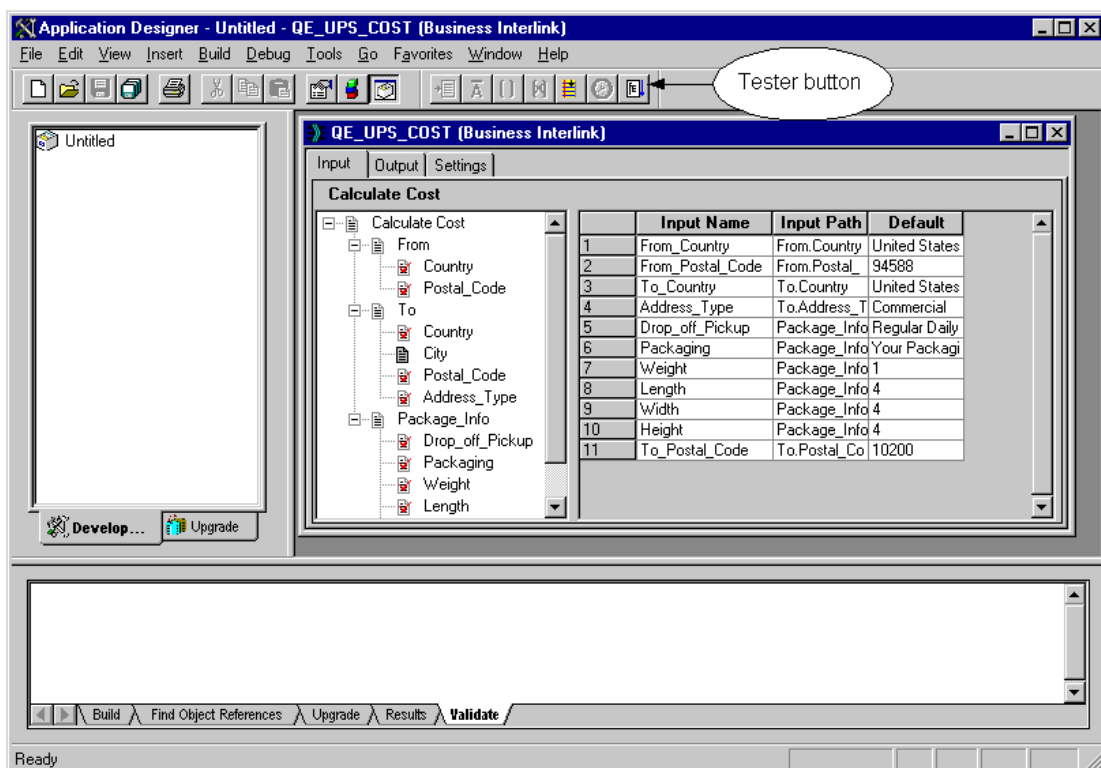
Business Interlink Page: Settings Tab

Business Interlink Tester Page: Testing Your Business Interlink Definition

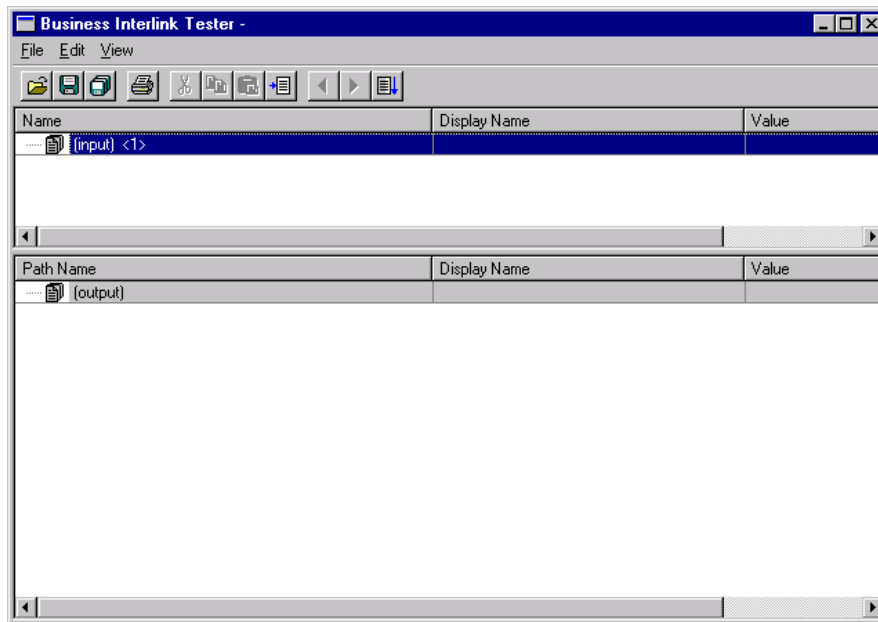
To run the Business Interlink Tester

1. Connect to any external system that your Business Interlink needs.
2. Start the Business Interlink Tester.

Open your Business Interlink (if it is not yet open) and either select **View:Test** or click the Tester button.

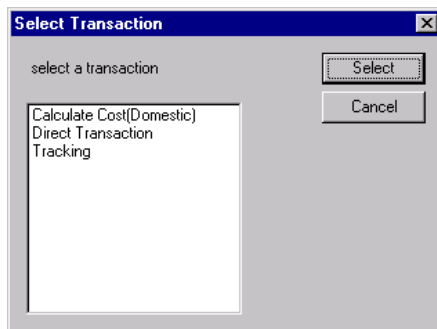


PeopleTools Application Designer Showing Tester Button



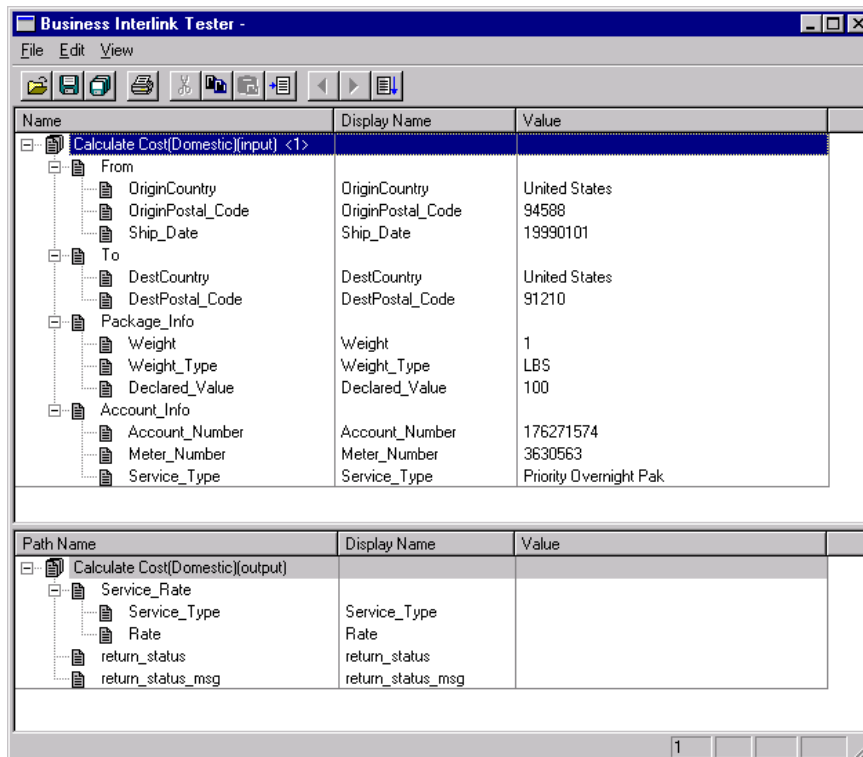
Business Interlink Tester, Started

3. From the Select Transaction pop-up window, select the Business Interlink transaction that you want to test. Once you select your transaction, the Business Interlink Tester will show all of the inputs and outputs for that transaction. This example tests the Calculate Cost(Domestic) transaction.






Business Interlink Tester: Select Transaction To Test

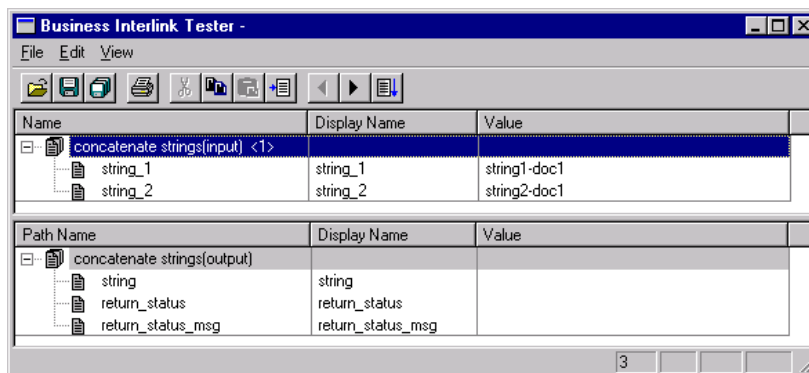
4. In the Business Interlink Tester page showing the inputs and outputs for your Business Interlink Transaction, enter input values for your transaction. If the XML design-time plug-in already defined values for them, they will appear in the value column. You can enter input values two ways:
 - Type the input values into the Value column.
 - If you have saved a set of input values as an Input Doc, select Open Input Doc, navigate to the location of your Input Doc, and then select the Input Doc. Step 5 shows how to save an Input Doc.



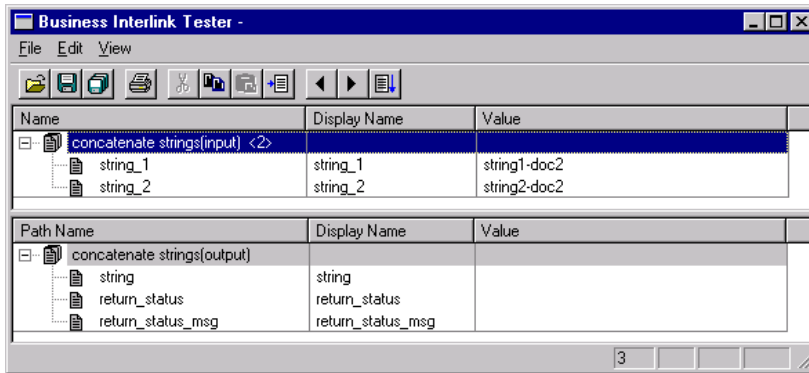
Business Interlink Tester: XML Design-Time Plug-In Opened, Input Values Entered

- If you wish to save your Input values, select **Save Input Doc** or **Save Input Doc As** from the File menu. You can then use those input values when you run the Business Interlink Tester on this transaction at a later time.
- You can enter more than one set of inputs, or input docs, to test, provided your Business Interlink plug-ins support it. To enter more than one input doc, highlight the root level of the input grid, click the Insert new doc button , and enter input values into  input doc. To move to each input doc, use the previous doc and next doc buttons .




The following example shows two input docs for a string concatenate Business Interlink.

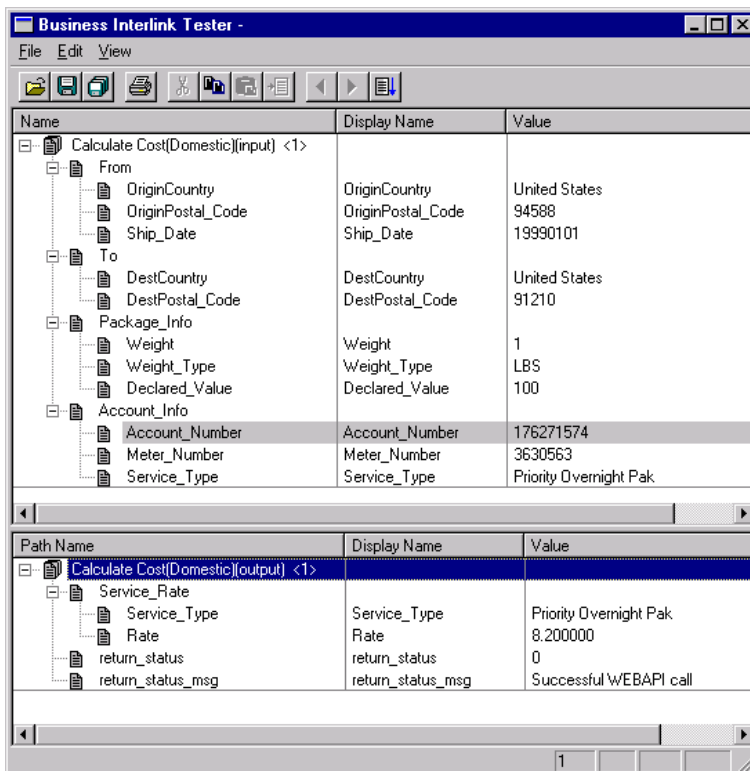


Business Interlink Tester: String Concatenate Input Doc 1



Business Interlink Tester: String Concatenate Input Doc 2

7. Once you have entered all the required input values, the Test button  is enabled. Press the Test button to test your Business Interlink plug-ins. Once the test has run, the bottom grid will contain a row of data for each output that you earlier defined under the Output tab. :If you entered multiple input docs and received multiple output docs as a result, view the multiple output docs by highlighting  at level of the output grid and clicking the previous doc and next doc buttons .






Business Interlink Tester, Output Values Received

8. If you want to print your inputs or outputs, highlight the inputs or outputs and then select **Print** from the File menu.

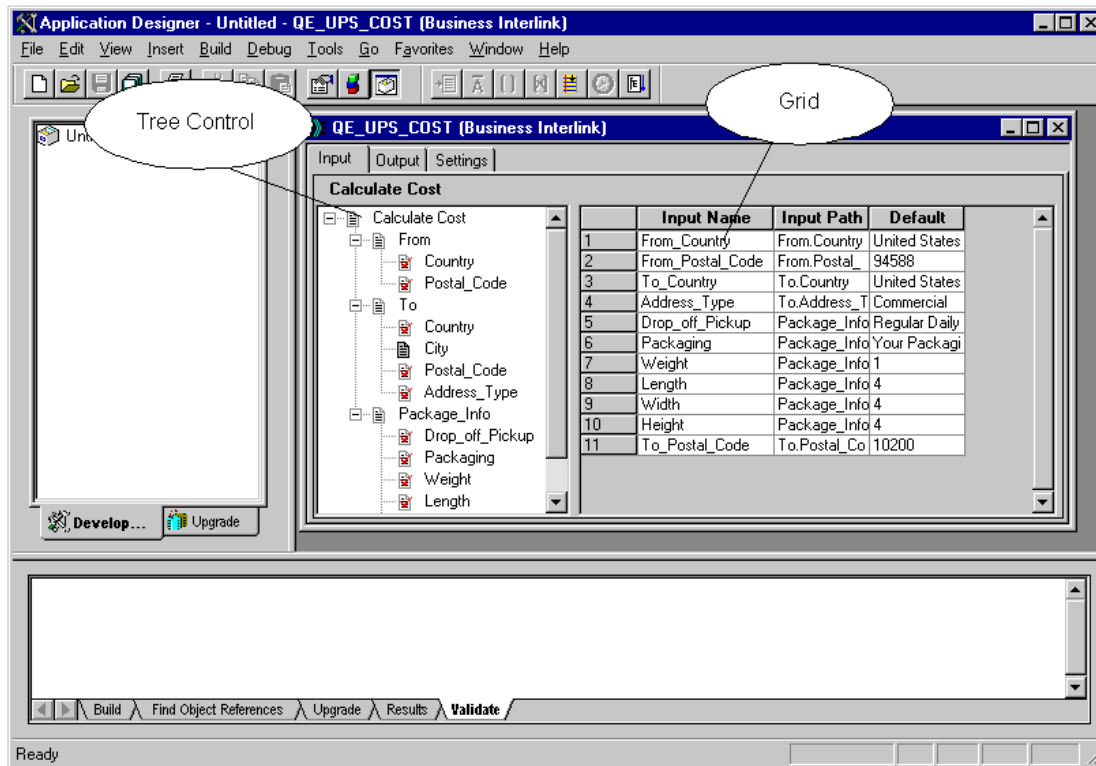
Using the Business Interlink Page

This section describes how to use the tree control and grid in the Business Interlink page. This page appears when you click the Input, Output, Criteria, or Settings tab, which you use to set the parameters for your transaction or query Business Interlink Definition. The Business Interlink page has a tree control on the left side of the page, and a grid on the right side. The tree control lists either transaction input/output parameters, class data members, or configuration parameters. You will select them for your Business Interlink Definition by performing any of the following actions:

- To expand the transaction, class, or object instances to show its inputs/outputs/data members, click the + to the left of the transaction/class/object instance.
- To select an input/output/data member for your Business Interlink Definition, drag it from the tree control and drop it on the grid. If you select an object (shown by a + or – to the left of its name), you select that object's OID for your Business Interlink Definition.
- Each row in the grid contains an input, output, or data member. To remove a row from the grid, highlight it by clicking on the corresponding number in the leftmost column of the grid, and either press Delete or select **File: Delete**.
- To add an empty row to the grid, highlight a row and click the Add Row button .
- To display the Data Type, Attribute (required status), and path information of an input/output/data member/configuration parameter, and the BulkExec ID for an input parameter, either right-click in the grid and select **More Info** from the pop-up menu, or click the Datatype button .
- To run the Business Interlink Tester after you have designed your Business Interlink, either select **View:Menu** or click the test button .



For more information about the Business Interlink Tester, see Business Interlink Tester Page: Testing Your Business Interlink Definition.

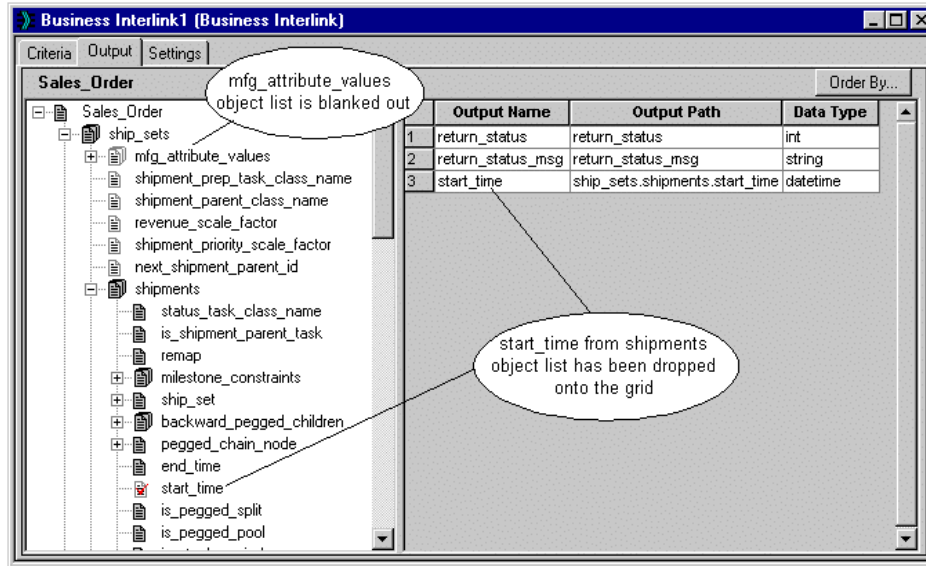


Business Interlink Definition Page Showing Tree Control and Grid

Supply Chain Restrictions On Drag And Drop From Control Tree

There are restrictions on the data members that you can drag and drop from the control tree to the grid. These restrictions are based on the data members that you have selected that are also lists. Lists are visible in the tree control by a multiple branch icon, such as is shown in the following figure. If a data member is made inaccessible by your previous selections, you will not be able to drag it from the tree control to the grid.

One restriction is that if you open a list of objects in the control tree, and then drag a data member from that object list to the grid, the other object lists in the control tree will be blanked out. Then you will not be able to open another object list at the same level or higher levels in the control tree and drag any of its data members to the grid. For example, where `mfg_attribute_values` and `shipments` are both object lists, you can drag a data member of `shipments` and drag it to the grid. But then `mfg_attribute_values` would be blanked out, and you would not be able to open it and drag any of its data members to the grid. This is shown below, where dragging and dropping the data member `start_time` from the `shipments` object list caused the `mfg_attribute_values` object list to blank out.



Control Tree with a Blanked Out Object List

Another restriction is at runtime; this restriction does not cause lists in the control tree to blank out, but can cause runtime errors. If you drag and drop a list to the grid, any other list that you drag and drop must have the same number of entries as the previously dropped list. At runtime, the Business Interlink Plug-in determines the number of entries in the list.

CHAPTER 3

Generating the PeopleCode Template

This section shows how to generate a Business Interlink PeopleCode template, using the Federal Express plug-in as an example.

You will use PeopleCode to run a Business Interlink Object, which is an instantiation of a Business Interlink Definition. You need to know how to enter the PeopleSoft Application Designer to use these instructions.

To run a Business Interlink Object for a Supply Chain Transaction or Supply Chain Query, you must start a Supply Chain server and rserver, or have ones available that you can use. Supply Chain needs this step when its transaction or query is executed; your external system might need a similar action in order to run its Business Interlink Objects.

You can run a Business Interlink Object as part of an online process (PeopleCode Event on a record, component, page, and so on) or a batch process (that is, an Application Engine program.) The example shows creating a program in the FieldChange event for a record field, but you could run it from any event or even as part of a message subscription.

Generating Business Interlink PeopleCode template in an Event

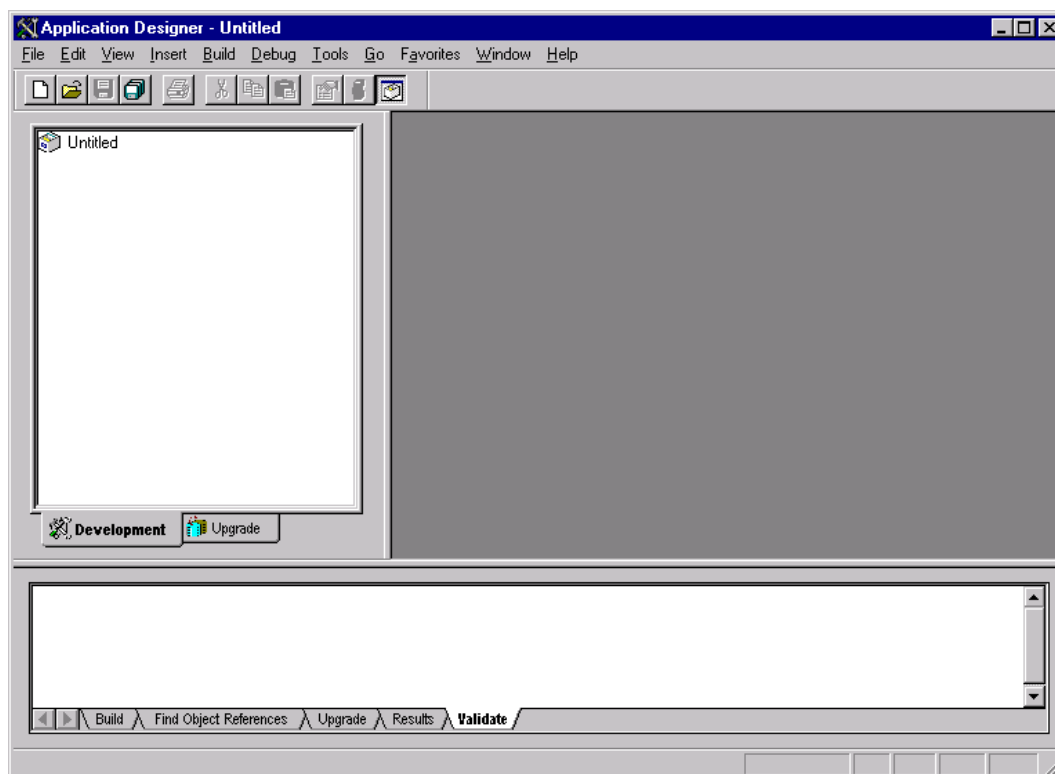


For more information about PeopleCode events, see PeopleCode and the Component Processor.

To generate a Business Interlink PeopleCode template in a FieldChange event

1. If you have not done so, start the PeopleSoft Application Designer.

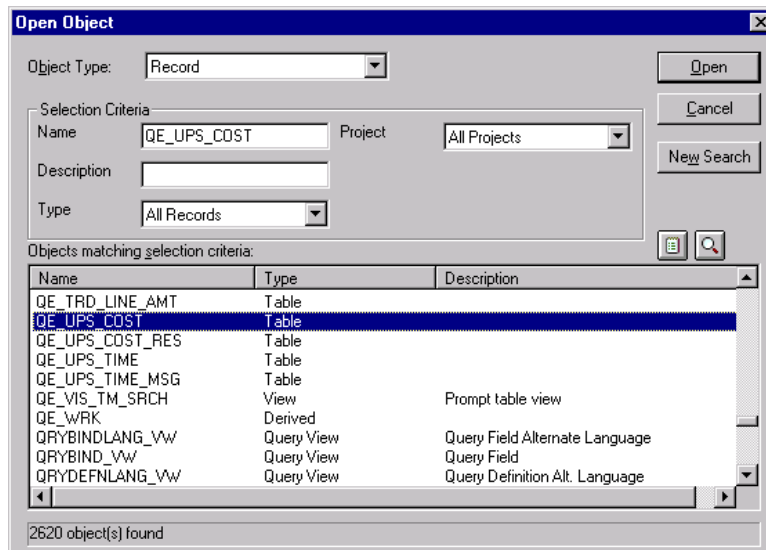
You will see a screen similar to the following.



PeopleSoft Application Designer


2. Select the PeopleSoft Record from which you will instantiate your Business Interlink Definition and execute the Business Interlink Object created by the instantiation.

Select **File, Open**. Then select **Record** from the Object Type drop-down list box in the Open Object dialog box to display the records, and press the Open button to list the records. In that list, double-click on the record you want to use. In this case, we are selecting the QE_UPS_COST record that is supplied with the Business Interlink software.



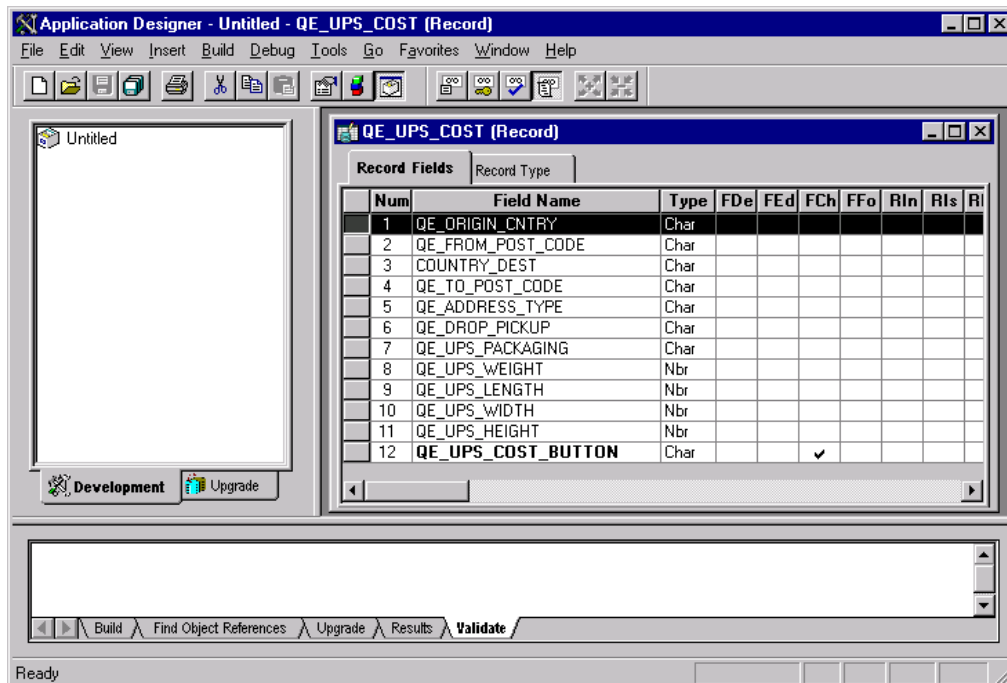
Opening a PeopleSoft Record

3. Choose the Record Field and the PeopleCode event from which you will instantiate and execute your Business Interlink Object.

Select **View, PeopleCode Display** or click the PeopleCode Display button  to display the PeopleCode events for the Record Fields. This is shown below.

From this page, decide which Record Field you want to enter your Business Interlink Object, then select the PeopleCode Event into which you want to instantiate and execute your Business Interlink Object by double-clicking on that Record Field's PeopleCode event. The page below shows that PeopleCode has already been entered for the QE_UPS_COST_BUTTON field on the FieldChange event, because that event contains a checkmark.

Once you double-click the event, the PeopleCode editor window for that Record Field's PeopleCode event appears. This page is shown after the following figure, which shows the PeopleCode Display page for the QE_UPS_COST record.



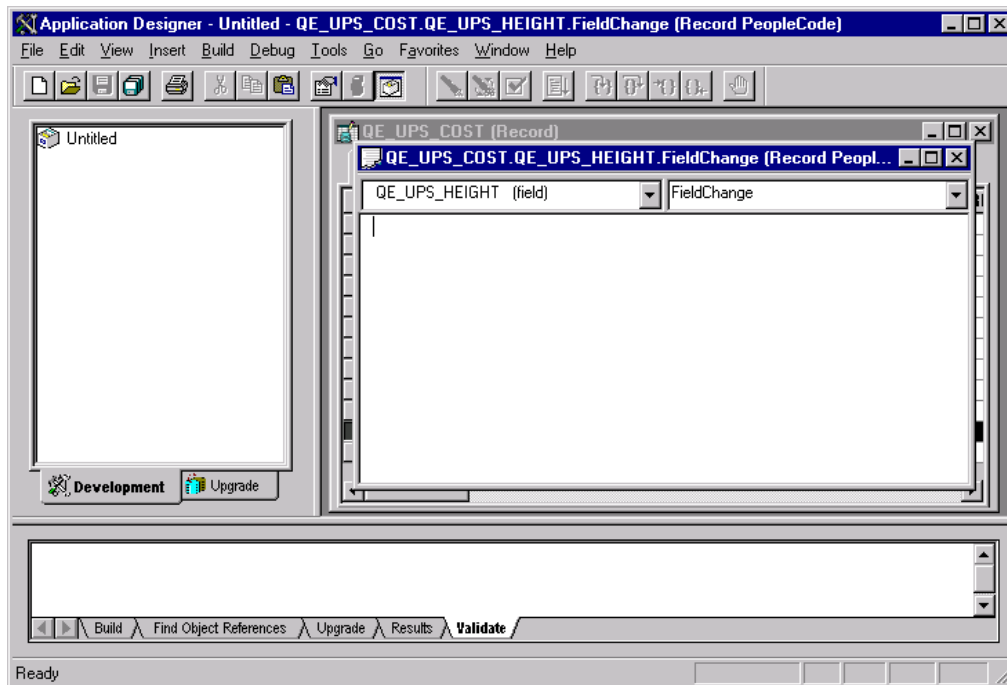
Record Fields, PeopleCode Events

The PeopleCode editor appears after you double-click the FieldChange checkbox.

PeopleCode is associated with one component and one event. Usually, the component is a Record Field, and the event is an action that an end-user takes upon a PeopleSoft page that references that Record Field. Such an event could be changing data in a Record Field displayed on the page (Field Change). It could also be saving the information on that page. The PeopleCode is executed when that event occurs. In this example, the PeopleCode is going to contain the code to execute the Business Interlink Object, which will be executed whenever the QE_UPS_HEIGHT field in the QE_UPS_COST record is changed in any page.



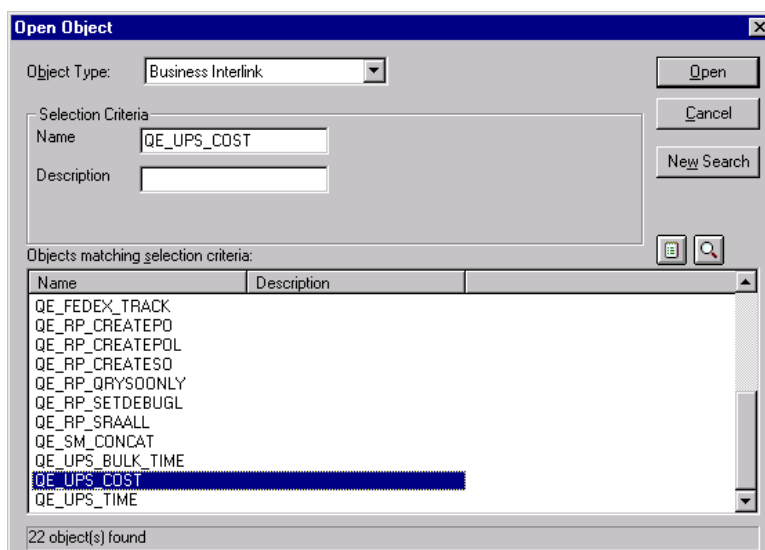
This example is from the QE_UPS_COST record, which has a PeopleCode event for the QE_UPS_COST_BUTTON field. The QE_UPS_HEIGHT field is being used in these steps only to show an empty PeopleCode editor and a Business Interlink PeopleCode template in the following steps.



PeopleCode Page, empty

4. Insert your Business Interlink Definition into the Project tree control on the left side of the Application Designer page.

There are several ways to do this. One way is to select **File, Open**. Then from the Open Object dialog box, select **Business Interlink** from the Object Type drop-down list box, click the Open button to get a list of Business Interlinks in the Objects matching selection criteria list, then double-click on the Business Interlink that you want to use. Then select **Insert, Current Object into Project**.



Insert into Project dialog box

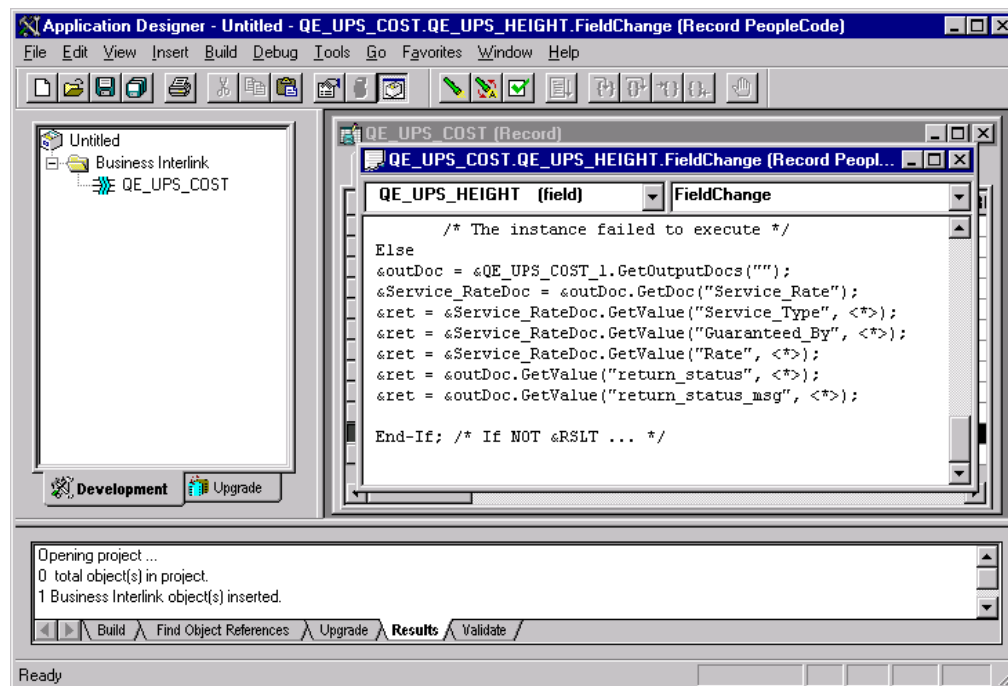
Another way is to select **Insert, Objects into Project**. Then from the Insert into Project dialog box, select **Business Interlink** from the Object Type drop-down list box, click the Insert button to get a list of Business Interlinks in the Objects matching selection criteria list, then double-click on the Business Interlink that you want to use.

If you have already inserted your Business Interlink into a project, you can select **File, Open**, select Project, click the Open button to list the projects, and select your project that contains this Business Interlink. That project, along with your Business Interlink, will then be displayed in the Project control tree.

5. Generate your PeopleCode template.

Generate a PeopleCode template in order to avoid a lot of typing. For example, the template contains calls to the Business Interlink Object methods, and includes the names of the inputs and outputs for your Business Interlink Object.

Generate the template by dragging and dropping your Business Interlink name from the Project tree control on the left side into the open PeopleCode editor window on the right side. A PeopleCode template is generated for your Business Interlink within the PeopleCode editor window.



PeopleCode editor Containing Template

6. Fill in the PeopleCode template so that this PeopleCode program can run your Business Interlink Object.



For instructions on filling the template, and for examples of PeopleCode templates, see [Running the Business Interlink Object](#).

Generating a Business Interlink PeopleCode Template For an Application Engine Program



For more information about Application Engine Programs, see Application Engine.

To generate a Business Interlink PeopleCode template for an Application Engine program

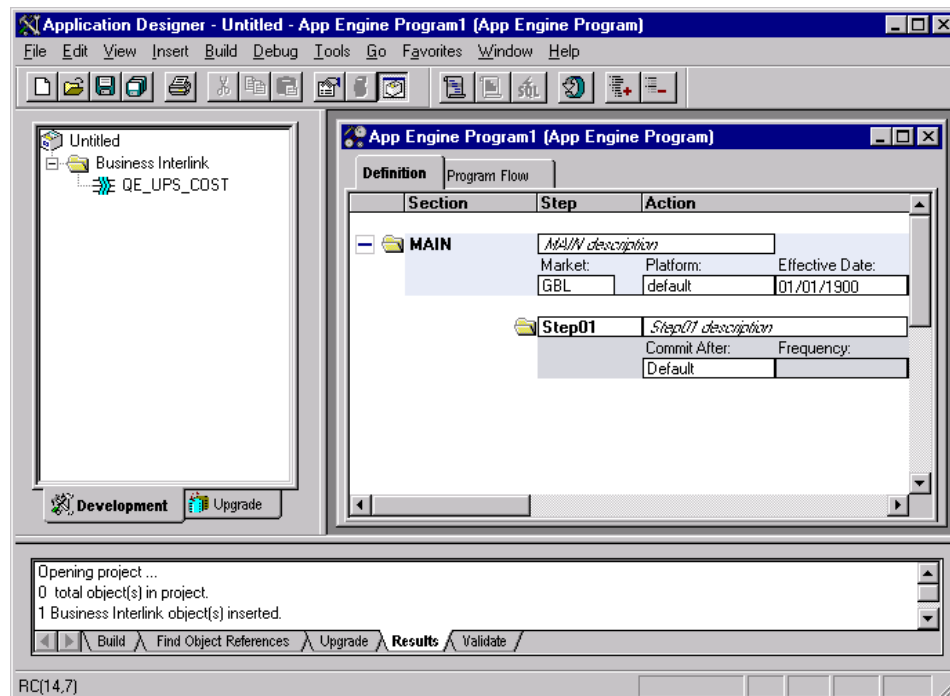
1. If you have not done so, start the PeopleSoft Application Designer.
2. Insert your Business Interlink Definition into the Project tree control on the left side of the Application Designer page.

For example, you can select **File, Open**, select a Project from the dialog box that contains the Business Interlink that you want to use with the Application Engine Program, then insert the Business Interlink. In the following step, **Insert, Objects into Project** is used to insert the Business Interlink.

3. Open or create an Application Engine Program.

For example, you can select **File, New**, and then select **App Engine Program** from the dialog box.

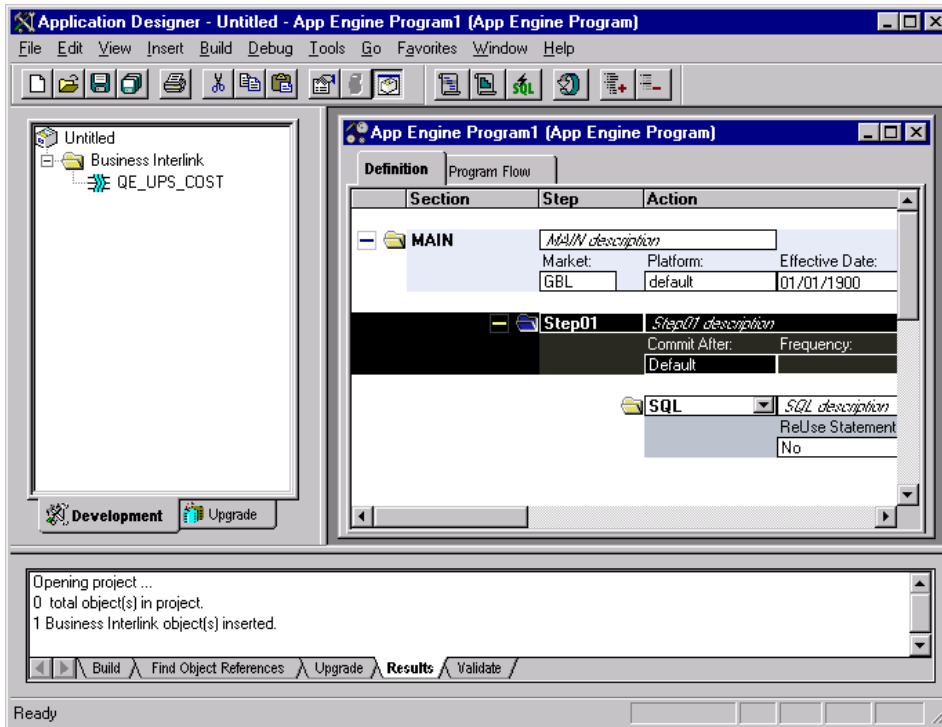
When you have opened the Application Engine Program, you get a screen similar to the following:



New Application Engine Program

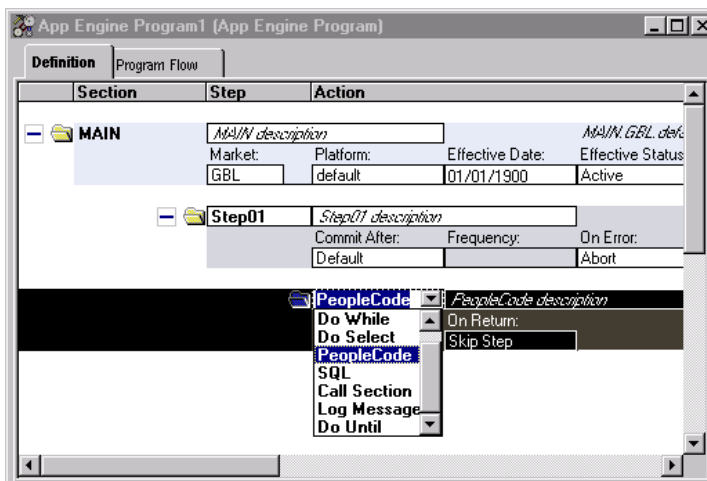
4. If this Application Engine Program does not have an action, or you want to create a new action for the PeopleCode program, select **Insert, Action**. In this case, you must highlight a step (Step01) before you can insert an action.

You will see the new action added.



Application Engine Program: Action Added

5. In the action, select PeopleCode from the drop-down menu.

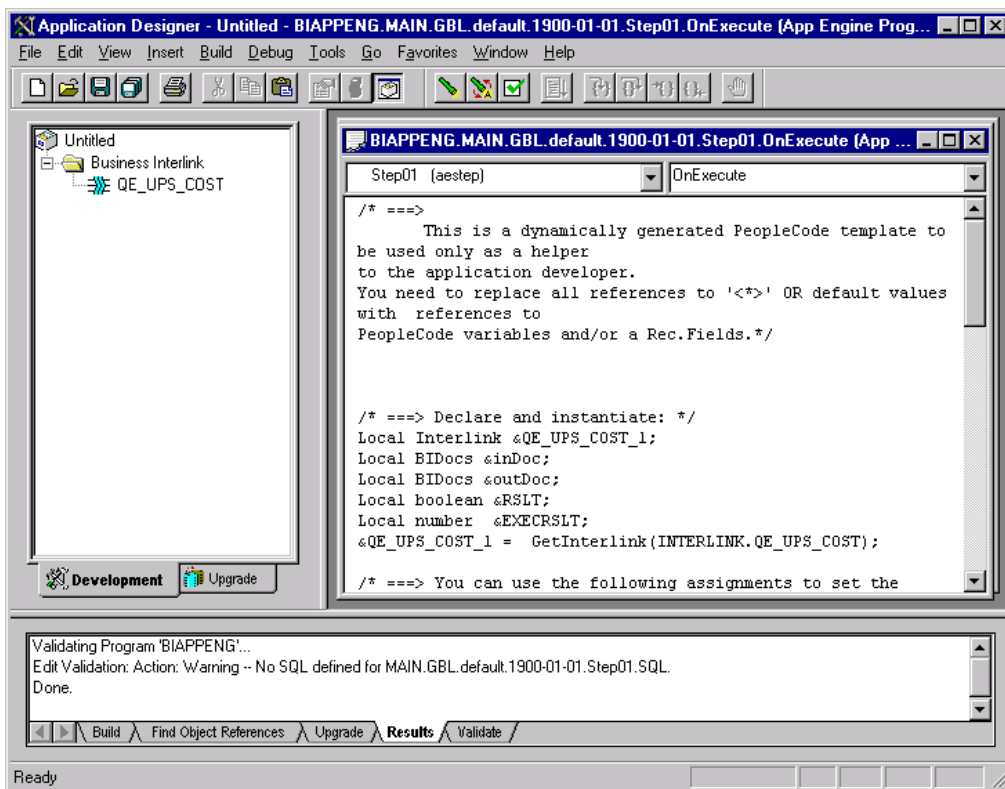


Application Engine Program: PeopleCode Action

6. If you have not done so, save the Application Engine Program. Then right-click on the PeopleCode action, and then select View PeopleCode from the popup. An empty PeopleCode editor window appears.
7. Generate your PeopleCode template.

Generate a PeopleCode template in order to avoid a lot of typing. For example, the template contains calls to the Business Interlink Object methods, and includes the names of the inputs and outputs for your Business Interlink Object.

Generate the template by dragging and dropping your Business Interlink name from the Project tree control on the left side into the open PeopleCode editor window on the right side. A PeopleCode template is generated for your Business Interlink within the PeopleCode editor window.



PeopleCode editor window Containing Template

8. Fill in the PeopleCode template so that this PeopleCode program can run your Business Interlink Object.



For instructions on filling the template, and for examples of PeopleCode templates, see Running the Business Interlink Object.

Running the Business Interlink Object

This section shows how to run a Business Interlink Object using PeopleCode, using a Federal Express Transaction as an example, along with some information about Supply Chain Queries.



For more information about PeopleCode, see PeopleCode Developer's Guide.

To run the Business Interlink Object, you write a PeopleCode program that executes the Business Interlink Object.

When you write a PeopleCode program, you fill in the PeopleCode template. (You could write the PeopleCode program from scratch, but it is easier to start with a template.) Filling in the PeopleCode template means writing a PeopleCode program that instantiates your Business Interlink Definition into a Business Interlink Object, and then executes the Business Interlink Object. Once this has been done, the Business Interlink Object is run whenever the PeopleCode program is executed. The PeopleCode program can be executed from an event or from an Application Engine Program, depending upon which you chose when you generated the PeopleCode template.



For more information on generation the PeopleCode template, see Generating the PeopleCode Template.

To fill in the PeopleCode template, you can replace all references to <*> with references to PeopleCode variables or to Record Fields, and by inserting values for the input parameters.

- For Business Interlink Object inputs, within the AddInputRow call, replace any references to <*> with a constant value, or a Record Field name, or a PeopleCode variable name.
- For Business Interlink Object outputs, within the FetchNextRow call, replace any references to <*> with a Record Field name or a PeopleCode variable name.
- There may be default values for the input parameters in the PeopleCode template. Optionally, you can replace default values for the input parameters by typing in new constants, Record Fields, or PeopleCode variables.

The PeopleCode template contains the code that you can start with after you dragged a Business Interlink Definition into a PeopleCode page. This PeopleCode template saves you a lot of typing by generating the main structure of PeopleCode to instantiate your Business Interlink Definition,

creating a Business Interlink Object, and then adding input values to, executing, and fetching output values from that Business Interlink Object.

The PeopleCode template consists of the following main sections:

- **Configuration Parameter settings.** If they have default values, you can type new values for them if you desire. Replace any <*> with a constant value, Record Field, or PeopleCode variable.
- **GetInterlink** method. This method instantiates your Business Interlink Definition, creating a Business Interlink Object that you can execute within PeopleCode.
- **AddDoc, AddValue** method. This section passes one set of input parameter values to the Business Interlink Object. You can use several methods to pass inputs.
 - The supplied **GetInputDocs, AddDoc, and AddValue** methods pass a row of input to the object. Replace each <*> with a constant value, Record Field, or PeopleCode variable; these are the input values that you pass to the transaction or query. If the inputs have default values, you can type new values for them if you desire. The input is stored in an input buffer containing one or more sets of input parameter values for this transaction or query. If you pass only one set of input parameter values, the transaction is executed in real time. If you place the **AddDoc** and **AddValue** methods in a loop (including the **AddNextDoc** method to add the next set of input parameters) and call them many times to add many sets of input parameter values to the input buffer, the transaction is executed in batch mode. Queries have one input row, and are executed in real time.
 - If your data is mapped to a rowset, you can replace the **GetInputDocs, AddDoc, and AddValue** methods with the **InputRowset** method. The **InputRowset** method will take a standard rowset object to populate the input buffers.
 - If you're sending a large amount of data to the input buffers, you might delete the supplied the **GetInputDocs, AddDoc, and AddValue** methods and write the data to a PeopleSoft record, then use the **BulkExecute** method. The **BulkExecute** method uses the data in the record to populate the input buffer, copying **like-named** fields. This method assumes that the names of the fields in the record match the names of the inputs defined in the Business Interlink Definition.
- **Execute** method. This section executes to the Business Interlink Object. You can use several methods to execute.
 - The supplied **Execute** method executes the transaction or query. You call this method once, and it will execute once for each row of input you have in the input buffer. (Since queries do not have multiple rows of input, they are executed once.)
 - If you're sending a large amount of data to the input buffers and you wrote the input data to a staging table, replace the **Execute** method with the **BulkExecute** method.
- **GetOutputDocs, GetDoc, and GetValue** methods. This section gets the outputs from the Business Interlink Object. You can use several methods to get the outputs.
 - The supplied **GetOutputDocs, GetDoc, and GetValue** methods get one set of output parameter values. For a transaction that was executed in batch mode (more than one set of input parameter values entered), you will usually get one set of output for each set of input.

Place the **GetDoc** and **GetValue** methods into a loop to do so, and include the **GetNextDoc** method to get the next set of output parameters. Replace each `<*>` with a Record Field or PeopleCode variable; these will be where the output values from the transaction or query are stored.

- If you used the **BulkExecute** method, it can automatically fill the output record specified with the method with the output values if you've specified an output record.
- If your data is hierarchical and you used the **InputRowset** method to take a standard rowset object to populate the input buffers, then replace the **GetOutputDocs**, **GetDoc**, and **GetValue** methods with the **FetchIntoRowset** method. The **FetchIntoRowset** method populates the rowset with data.
- If the output data is dynamic, meaning that the outputs for the Business Interlink object are changeable in data type or number of outputs, then you can use the **MoveFirst** and **MoveNext**, methods to loop through each row of the output buffer, use the **GetFieldCount** method to loop through the columns in the output buffer (which gets each output in the row), and use the **GetFieldName**, **GetFieldValue**, and **GetFieldType** methods to get the name, value, and type of each output.



For more information about these methods, refer to Using the Business Interlink Object Methods.

Below is code that is generated for sample Business Interlink Definitions.

Example of a transaction PeopleCode template

When you drag over the Business Interlink Definition QE_UPS_COST, which is for the transaction Calculate Cost, you get the following PeopleCode. This Business Interlink Definition is supplied with the Business Interlink software.

In addition to the template code and comments, this example includes some additional comments, marked with the text "ADDITIONAL COMMENT".

```
/* ===>
```

```
    This is a dynamically generated PeopleCode template to be used only as a
    helper to the application developer.
```

```
    You need to replace all references to '<*>' OR default values with references to
    PeopleCode variables and/or a Rec.Fields.*/
```

```
/* ===> Declare and instantiate: */
```

```

Local Interlink &QE_UPS_COST_1;

Local BIDocs &inDoc;

Local BIDocs &outDoc;

Local boolean &RSLT;

Local number &EXECRSLT;

&QE_UPS_COST_1 = GetInterlink(INTERLINK.QE_UPS_COST);

/* ==> You can use the following assignments to set the configuration
parameters.

*/

/* ADDITIONAL COMMENT: The URL parameter points to the Business Interlink
runtime plug-in.*/

&QE_UPS_COST_1.URL = file://ups.dll;

/* ==> You might want to call the following statement in a loop if you have
more than one row of data to be added. */

/* ADDITIONAL COMMENT: The AddValue calls use for its inputs the input names in
the Input Path column of the Input Tab for this Business Interlink Definition.

The AddValue calls also contain all of the default values that have been set for
the QE_UPS_COST Business Interlink Definition. You will likely replace them with
variables or record fields when you complete the coding for this template. */

/* ==> Add inputs: */

&inDoc = &QE_UPS_COST_1.GetInputDocs("");

&FromDoc = &inDoc.AddDoc("From");

&ret = &FromDoc.AddValue("Country", <*>);

&ret = &FromDoc.AddValue("Postal_Code", <*>);

&ToDoc = &inDoc.AddDoc("To");

```

```

&ret = &ToDoc.AddValue("Country", <*>);

&ret = &ToDoc.AddValue("Address_Type", <*>);

&ret = &ToDoc.AddValue("Postal_Code", <*>);

&Package_InfoDoc = &inDoc.AddDoc("Package_Info");

&ret = &Package_InfoDoc.AddValue("Drop_off_Pickup", <*>);

&ret = &Package_InfoDoc.AddValue("Packaging", <*>);

&ret = &Package_InfoDoc.AddValue("Weight", <*>);

&ret = &Package_InfoDoc.AddValue("Length", <*>);

&ret = &Package_InfoDoc.AddValue("Width", <*>);

&ret = &Package_InfoDoc.AddValue("Height", <*>);


/* ==> The following statement executes this instance: */

&EXECRSLT = &QE_UPS_COST_1.Execute();

If ( &EXECRSLT <> 1 ) Then

    /* The instance failed to execute */

Else

    /* ADDITIONAL COMMENT: The GetValue calls use for the outputs the output names
    in the Output Path column of the Output Tab for this Business Interlink
    Definition. */

    &outDoc = &QE_UPS_COST_1.GetOutputDocs("");

    &Service_RateDoc = &outDoc.GetDoc("Service_Rate");

    &ret = &Service_RateDoc.GetValue("Service_Type", <*>);

    &ret = &Service_RateDoc.GetValue("Guaranteed_By", <*>);

    &ret = &Service_RateDoc.GetValue("Rate", <*>);

    &ret = &outDoc.GetValue("return_status", <*>);

    &ret = &outDoc.GetValue("return_status_msg", <*>);

End-If; /* If NOT &RSLT ... */

```

Example of a transaction PeopleCode template having been fully coded

The following code is an example of how the template for the Business Interlink Definition QE_UPS_COST could be coded. This PeopleCode is an edited version of the PeopleCode contained within the QE_UPS_COST record, QE_UPS_COST_BUTTON field, FieldChange PeopleCode event.

This code executes the transaction named Calculate Cost(Domestic), which calculates the cost of shipping a package. It also execute the Tracking transaction, which tracks a package.

Following this code is a diagram of the inputs and outputs for these transactions, and then a short discussion of the records where this PeopleCode is used.

In addition to the code and comments in this PeopleCode event, this example includes comments pointing out the changes made to the template. These comments are marked with the text "CHANGE".

```

/* ==> Declare and instantiate: */

Local Interlink &QE_UPS_COST_1;

Local boolean &RSLT;

Local number &EXECSRSLT;

Local Record &REC;

Local number &count;

Local BIDocs &biDocs, &InputDocs;

Local BIDocs &ServiceRateDoc, &FromDoc, &ToDoc, &PackageDoc;

Local number &ret;

&QE_UPS_COST_1 = GetInterlink(Interlink.QE_UPS_COST);

/* ==> You can use the following assignments to set the configuration
parameters.

*/

&QE_UPS_COST_1.URL = "file://ups.dll";

GetLevel0() (1).GetRowset(Scroll.QE_UPS_COST_RES).Flush();

```

```

/* ==> You might want to call the following statement in a loop if you
have more than one row of data to be added. */

/* ==> Add inputs: */

/* CHANGE: The input values are set to fields within the QE_UPS_COST record,
rather than to default values. */

&REC = GetLevel0().GetRow(1).GetRecord(Record.QE_UPS_COST);

&COUNTRY_FROM = &REC.QE_ORIGIN_CNTRY.Value;

&FROM_POST_CODE = &REC.QE_FROM_POST_CODE.Value;

&COUNTRY_DEST = &REC.COUNTRY_DEST.Value;

&TO_POSTAL_CODE = &REC.QE_TO_POST_CODE.Value;

&DROP_OFF_PICKUP = &REC.QE_DROP_PICKUP.Value;

&PACKAGING = &REC.QE_UPS_PACKAGING.Value;

&WEIGHT = &REC.QE_UPS_WEIGHT.Value;

&LENGTH = &REC.QE_UPS_LENGTH.Value;

&WIDTH = &REC.QE_UPS_WIDTH.Value;

&HEIGHT = &REC.QE_UPS_HEIGHT.Value;

/* &DESTINATION = &REC.QE_TO_ZIP.Value; */

/* CHANGE: Evaluate several of the parameters */

/* Fix up the County From data */

Evaluate &COUNTRY_FROM

When = "USA"

    &COUNTRY_FROM = "United States"

End-Evaluate;

/* Fix up the County dest data */

Evaluate &COUNTRY_DEST

When = "USA"

```

```
&COUNTRY_DEST = "United States"

When = "UK"

    &COUNTRY_DEST = "United Kingdom"

When = "CAN"

    &COUNTRY_DEST = "Canada"

End-Evaluate;


/* Evaluate the translates */

Evaluate &DROP_OFF_PICKUP

When = "A"

    &DROP_OFF_PICKUP = "On Call Air";

When = "L"

    &DROP_OFF_PICKUP = "Letter Center";

When = "O"

    &DROP_OFF_PICKUP = "One Time Pickup";

When = "R"

    &DROP_OFF_PICKUP = "Regular Daily Pickup";

End-Evaluate;


Evaluate &PACKAGING

When = "E"

    &PACKAGING = "UPS Express Box";

When = "L"

    &PACKAGING = "UPS Letter Envelope";

When = "T"

    &PACKAGING = "UPS Tube";

When = "Y"

    &PACKAGING = "Your Packaging";

End-Evaluate;
```

```

/* New Hierarchical data buffer stuff */

/* CHANGE: The <*> in the AddValue methods are replaced with proper values. */
&InputDocs = &QE_UPS_COST_1.GetInputDocs("");
&FromDoc = &InputDocs.AddDoc("From");
&ret = &FromDoc.AddValue("Country", "United States");
&ret = &FromDoc.AddValue("Postal_Code", &FROM_POST_CODE);

&ToDoc = &InputDocs.AddDoc("To");
&ret = &ToDoc.AddValue("Country", &COUNTRY_DEST);
&ret = &ToDoc.AddValue("Postal_Code", &TO_POSTAL_CODE);
&ret = &ToDoc.AddValue("Address_Type", "Commercial");

&PackageDoc = &InputDocs.AddDoc("Package_Info");
&ret = &PackageDoc.AddValue("Drop_off_Pickup", &DROP_OFF_PICKUP);
&ret = &PackageDoc.AddValue("Packaging", &PACKAGING);
&ret = &PackageDoc.AddValue("Weight", &WEIGHT);
&ret = &PackageDoc.AddValue("Length", &LENGTH);
&ret = &PackageDoc.AddValue("Width", &WIDTH);
&ret = &PackageDoc.AddValue("Height", &HEIGHT);

/* ==> The following statement executes this instance: */
&EXECRSLT = &QE_UPS_COST_1.Execute();

If (&EXECRSLT <> 1) Then

    /* The instance failed to execute */

Else

```

```

/* ==> Fetch Outputs: */

/* CHANGE: The <*> in the AddValue methods are replaced with proper values. */

&RSLT = True;

&ROWSET = GetRowset(Record.QE_UPS_COST_RES);

&I = 1;

&biDocs = &QE_UPS_COST_1.GetOutputDocs("");

/* CHANGE: Because Service_Rate is a list, you must get its values using a loop.
GetCount gets the number of Service_Rate output parameters. */

/* Possible fix: the following line may need to be added here:

&ServiceRateDoc = &biDocs.GetDoc("Service_Rate"); */

/* Possible fix: the following line may need to be commented out */

&count = &biDocs.GetCount("Service_Rate");

While (&I < &count)

    &ServiceRateDoc = &biDocs.GetDoc("Service_Rate");

    &ret = &ServiceRateDoc.GetValue("Service_Type", &SERVICE_TYPE);

    &ret = &ServiceRateDoc.GetValue("Guaranteed_By", &GUAR_BY);

    &ret = &ServiceRateDoc.GetValue("Rate", &RATE);

    If &ret = 0 Then

        If &I > 1 Then

            &ROWSET.InsertRow(&I - 1);

        End-If;

        UpdateValue(QE_UPS_COST_RES.QE_UPS_SVC_TYPE, &I, &SERVICE_TYPE);

        UpdateValue(QE_UPS_COST_RES.QE_UPS_GUAR_BY, &I, &GUAR_BY);

        UpdateValue(QE_UPS_COST_RES.QE_UPS_RATE, &I, &RATE);

        &I = &I + 1;

    /* Possible fix: the following line may need to be added here:

        &ret = &ServiceRateDoc.GetNextDoc(); */

    End-If;

End-While;

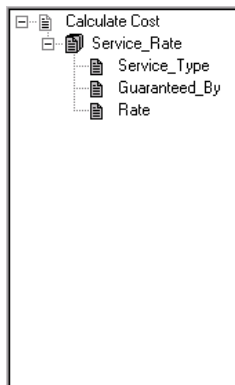
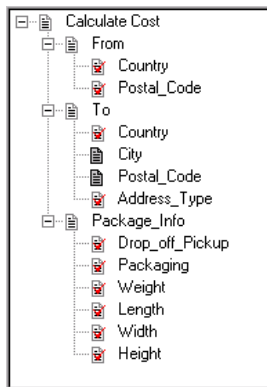
```



```
End-If; /* If NOT &RSLT ... */
```

Diagrams of the Inputs and Outputs for the Transactions

Here are diagrams showing the structure of the inputs and outputs for the Calculate Cost transaction.



Inputs and Outputs for Calculate Cost(Domestic) Transaction

Page for the fully coded transaction PeopleCode template

Here is the page in which the PeopleCode event is used. This is a page named QE_UPS_TIME_PNL.

Business Interlink QA Page

The QE_UPS_COST_BUTTON field has the previous PeopleCode program in the FieldChange PeopleCode event, and that field is attached to the Execute button in this page. When a user clicks the Execute button, the FieldChange PeopleCode event is executed.

The inputs are taken mostly from the QE_UPS_COST fields in this page.

The fields for this page are shown below. Most of these fields are from the QE_UPS_COST record.

QE_UPS_TIME_PNL.ENG (Page)								
Page Designer		Order						
	Lvl	Label	Type	Field	Record	Display Control	Related Field	
1	0	UPS Time-in-Tran	Group Box		QE_UPS_TIME			
2	0	UPS QuickCost te	Group Box		QE_UPS_TIME			
3	0	Static Image	Static Image					
4	0	Origin	Edit Box	QE_FROM_ZIP	QE_UPS_TIME			
5	0	Destination	Edit Box	QE_TO_ZIP	QE_UPS_TIME			
6	0	Calc Time	Push Button/Hyp	QE_UPS_TIME_	QE_UPS_TIME			
7	0	Origin Country	Drop Down List	QE_ORIGIN_CNT	QE_UPS_COST			
8	0	Postal Code	Edit Box	QE_FROM_POST	QE_UPS_COST			
9	0	Dest Country	Drop Down List	COUNTRY_DES	QE_UPS_COST			
10	0	Postal Code	Edit Box	QE_TO_POST_C	QE_UPS_COST			
11	0	Package Informati	Group Box		QE_UPS_TIME			
12	0	Drop off/Pickup	Drop Down List	QE_DROP_PICK	QE_UPS_COST			
13	0	Packaging	Drop Down List	QE_UPS_PACKA	QE_UPS_COST			
14	0	Weight	Edit Box	QE_UPS_WEIGH	QE_UPS_COST			
15	0	Length	Edit Box	QE_UPS LENGT	QE_UPS_COST			
16	0	Width	Edit Box	QE_UPS_WIDTH	QE_UPS_COST			
17	0	Height	Edit Box	QE_UPS_HEIGH	QE_UPS_COST			
18	0	Calc Cost	Push Button/Hyp	QE_UPS_COST_	QE_UPS_COST			
19	0	Static Image	Static Image					
20	0	Ground Time-in-Tr	Long Edit Box	DESCRLONG	QE_UPS_TIME			
21	1	Dummy Name	Grid		QE_UPS_COST_RES			

Business Interlink QA Page Fields

Example: PeopleCode template for query sale_orders

When you drag over the Business Interlink Definition for a query on the class sales_orders, an example of which is shown in Designing a Supply Chain Business Interlink Definition, you get the following PeopleCode. The outputs for this query are site.display_name and display_name; since the input name for site.display_name was display_name, the creator of this Business Interlink edited the name to be site_display_name.

Notice that this query PeopleCode template is very similar to a transaction PeopleCode template. As with a transaction, you will fill in the inputs and outputs. Notice that the code does not show you how the query itself is built; use the Criteria tab in the Application Designer to see how the query is built.

In the case of this query, you will have one input row, so you will call AddInputRow once. And you will have many rows of output, so you will call FetchNextRow many times and then map the outputs to PeopleCode variables and/or record fields. Instead of calling FetchNextRow and mapping, you could instead call FetchIntoRecord once to copy the entire output buffer of this Business Interlink Object into a PeopleSoft record.



For more information about FetchIntoRecord, see FetchIntoRecord.

```
/* ==>
```

```
This is a dynamically generated PeopleCode template to be used only as a helper
to the application developer.
```

```
You need to replace all references to '<*>' OR default values with references to
PeopleCode variables and/or a Rec.Fields.*/
```

```
/* ==> Declare and instantiate: */
```

```
Local Interlink &QE_RP_QRYSOON_1;
```

```
Local boolean &RSLT;
```

```
Local number &EXECSRSLT;
```

```
&QE_RP_QRYSOON_1 = GetInterlink(INTERLINK.QE_RP_QRYSOONLY_01);
```

```
/* ==> You can use the following assignments to set the configuration
parameters.
```

```
*/
```

```

&QE_RP_QRYSOON_1.URL = "file://";

&QE_RP_QRYSOON_1.SERVER_NAME = "server";

&QE_RP_QRYSOON_1.RSERVER_HOST = "reddock";

&QE_RP_QRYSOON_1.RSERVER_PORT = "9884";

&QE_RP_QRYSOON_1.TIMEOUT = <*>;

/* ===> You might want to call the following statement in a loop if you have
more than one row of data to be added. */

/* ===> Add Criteria inputs (RHS): */

&QE_RP_QRYSOON_1.AddInputRow("IN1_display_name", <*> , "IN2_order_number", <*>
, "IN3_order_date", <*> , "IN4_order_date", <*> );

/* ===> The following statement executes this instance: */

&EXECRLT = &QE_RP_QRYSOON_1.Execute();

If ( &EXECRLT <> 1 ) Then

    /* The instance failed to execute */

Else

/* ===> Fetch Outputs: */

&RSLT = true;

while &RSLT

    &RSLT = &QE_RP_QRYSOON_1.FetchNextRow("cust_display_name", <*>,
"order_date", <*>, "site_display_name", <*>, "display_name", <*>,
"return_status", <*>, "return_status_msg", <*> );

/* Some processing based on the fetched values ... */

end-while;

End-If; /* If NOT &RSLT ... */

```

CHAPTER 5

Creating an Incoming Business Interlink

This section lays out the steps you follow to create an Incoming Business Interlink. An Incoming Business Interlink allows an IScript function (PeopleCode program) to receive an XML request via HTTP from a third party system, and send an XML response to the third party system.

Requirements to determine if Incoming Business Interlink suits your needs

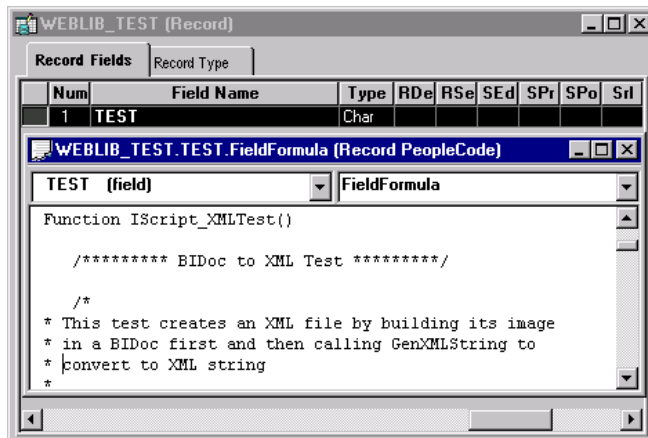
If you want to send your XML request to PeopleSoft, and receive XML responses, your transactions must be:

- Sessionless.
- Synchronized.
- Over the internet (HTTP).

Setting up the Incoming Business Interlink Environment for an IScript Function

To set up the environment for the IScript function (PeopleCode program) that performs Incoming Business Interlink functionality

1. Open or create a record named `WEBLIB_name`, where *name* is the name that you want for your IScript record.
2. Within that record, select the FieldFormula field (FFo). Write your IScript function (PeopleCode program) within this field. The following shows the IScript function named `IScript_Test`, contained in the `WEBLIB_test` record, its TEST field containing the IScript function within FieldFormula.

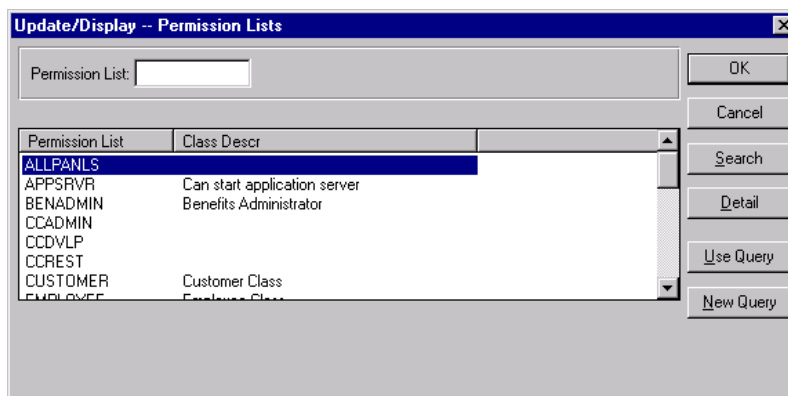


IScript Function in Test Field: Field Formula



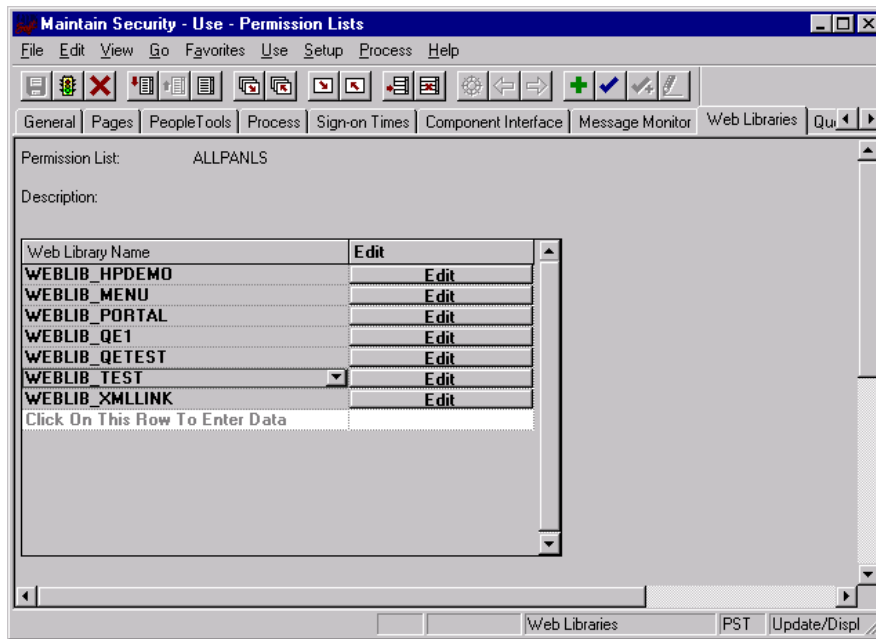
For more information about writing the PeopleCode program, see Writing the IScript Function.

3. Register the IScript function by service name (steps 3 through 7). Within the Application Designer page, select **Go, PeopleTools, Maintain Security**.
4. Within the Maintain Security page, select Use, Permission Lists, Web Libraries, Update/Display.
5. Within the Update/Display -- Permission Lists page, click OK, then double-click on the type of permission that you want from the Permission List listbox. For example, ALLPANLS provides the most generic permissions.



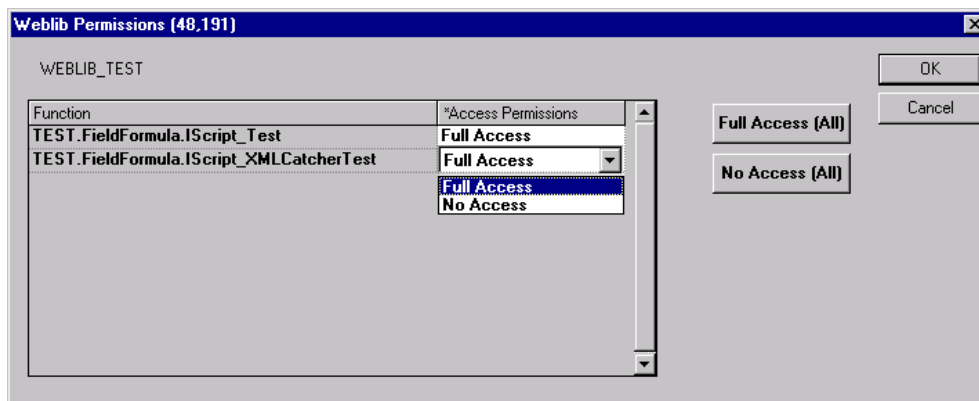
Update/Display -- Permission Lists

6. Within the Permission Lists page, click on the Edit button next to your WEBLIB_name. If your WEBLIB_name does not appear in this panel, add it by clicking on **Click On This Row To Enter Data** and selecting your WEBLIB_name from the drop-down list.



Maintain Security -- Permission Lists

7. Associate a service name to the IScript function (steps 7 through 9). Within the WebLib Permissions page, select **Full Access (All)** for your IScript and click OK.



Maintain Security -- Permission Lists

8. Within the Application Designer, select **Go, PeopleTools, Utilities**. Within the Utilities page, select **Use, XML Link Function Registry, Update/Display** and click OK.
9. Within the Utilities – Use – XML Link Function Registry page, enter the name you want for your service. Then, in the following order, select from the drop-down lists, and then select Save:
 - a. Record: Enter the name of the record containing your IScript function (PeopleCode program).
 - b. Field: Enter the name of the record field.

- c. Event: Select Field Formula.
- d. Function Name: Select your IScript function name.

Utilities – Use – XML Link Function Registry

Test your creation using the XMLLink tester

Access the xmllink servlet and check if your service appears in the list. Using the browser, get the URL `http://host/servlet/xmllink/PIApath`, where *host* is your server name, and *PIApath* is the default path for PIA (this is usually peoplesoft8). This will provide the list of registered services.

Test your set-up. For example, you can do a POST to the following URL:

```
http://host/servlet/xmllink/PIApath/servicename?userid=PTDMO&pwd=PTDMO)
```

where *host* is your server name, *PIApath* is the default path for PIA (this is usually peoplesoft8), and PTDMO is the username, and the password. *servicename* is the name of the service: in this example, that is SkillsSearch.

You can create a Business Interlink pshttpenable design-time plug-in to send an XML request and receive the XML response.



For more information about a design-time plug-in that can be used to test the example provided in this documentation, see Test Example: XML Design-Time Plug-in Used with an Incoming Business Interlink.



For more information about writing the pshttpenable design-time plug-in in general, see Writing a XML Design-Time Plug-In using the pshttpenable Runtime Plug-In.

Writing the IScript Function

The IScript function that you write performs the following tasks:

- Navigates the request XML using either the new BiDoc methods (listed at below) or populating data on an existing Business Interlink BiDoc using the fromXML(string) function.
- Does back-end processing to get data for the response.
- Creates the response XML using the new BiDocs methods (or populate an existing Business Interlink with addValue(string)).

Here is a PeopleCode program that performs these tasks for the Incoming Business Interlink example in this chapter.

```

/***** Locals *****/

Local BIDocs &rootDoc, &rootInDoc;

Local BIDocs &postregresponse;

Local BIDocs &error, &errorcode, &errortext;

Local BIDocs &candidates;

Local BIDocs &user, &conid, &username, &comments, &role, &availability, &rate,
&relevancy, &hobbies;

Local BIDocs &location, &city, &state, &zip, &country;

Local BIDocs &desiredloc;

Local string &xmlString;

Local SQL &GetCandidatesSQL;

Function IScript_Test()

    /***** BiDoc to XML Test *****/

    /*

```

```

* This test creates an XML file by building its image
* in a BiDoc and then calling GenXMLString to convert to XML string
*
*/

/***** Main *****/

/* Receive the request object and convert to an XML string */
&blob = %Request.GetContentBody();

/* process the incoming xml(request)- Create a BiDoc*/
&rootInDoc = GetBiDoc(&blob);

/* Use BiDocs methods to extract the skills value from this XML
   request. */
&postreqDoc = &rootInDoc.GetDoc("postreq");
&ret = &postreqDoc.GetValue("skills", &skills);

/*create SQL object to query the Database*/

&GetCandidatesSQL = CreateSQL("SELECT CONID, SKILLS, USERNAME, COMMENTS,
SCENERY, DENSITY, BLANK, CITY, STATE, ZIP, COUNTRY, DESIREDLOC, ROLE,
AVAILABILITY, RATE, RELEVANCY, RATING, HOBBIES FROM PS_CANDIDATES WHERE SKILLS
=:1 ORDER BY CONID", &skills);

/* Create the BiDoc which will have the same structure as the XML
   response*/
&rootDoc = GetBiDoc("");

/* add a processing instruction (the first line in the XML
   response) */
&ret = &rootDoc.addProcessInstruction("<?xml version=\"1.0\"?>");

```

```

/* create the postreqresponse tag for the XML response */

&postreqresponse = &rootDoc.createElement("postreqresponse");

/* create the error tag within the postreqresponse tag */

&error = &postreqresponse.createElement("error");

/* create the errorcode and errortext tags within the error tag
*/

&errorcode = &error.createElement("errorcode");

&ret = &errorcode.addText("1");

&errortext = &error.createElement("errortext");

/* create the candidates tag within the postreqresponse tag */

&candidates = &postreqresponse.createElement("candidates");

/* Fetch all the rows for users and create and populate the
   candidates tag with user tags and their data dynamically*/

While (&GetCandidatesSQL.Fetch(&conidvalue, &skillsvalue, &usernamevalue,
&commentsvalue, &sceneryvalue, &densityvalue, &blankvalue, &cityvalue,
&statevalue, &zipvalue, &countryvalue, &desiredlocvalue, &rolevalue,
&availabilityvalue, &ratevalue, &relevancyvalue, &ratingvalue, &hobbiesvalue))

    &user = &candidates.createElement("user");

    /* create the tags within the user tag to contain data */

    &conid = &user.createElement("conid");

    &ret = &conid.addText(&conidvalue);

    &username = &user.createElement("username");

    &ret = &username.addText(&usernamevalue);

    &comments = &user.createElement("comments");

    &ret = &comments.addText(&commentsvalue);

    &location = &user.createElement("location");

```

```
/* Add some attributes to the location tag*/

&ret = &location.addAttribute("scenery", &sceneryvalue);

&ret = &location.addAttribute("density", &densityvalue);

&ret = &location.addAttribute("blank", &blankvalue);


/* Add tags and their data to the location tag */

&city = &location.createElement("city");

&ret = &city.addText(&cityvalue);

&state = &location.createElement("state");

&ret = &state.addText(&statevalue);

&zip = &location.createElement("zip");

&ret = &zip.addText(&zipvalue);

&country = &location.createElement("country");

&ret = &country.addText(&countryvalue);

&desiredloc = &user.createElement("desiredloc");

&ret = &desiredloc.addText(&desiredlocvalue);

&role = &user.createElement("role");

&ret = &role.addText(&rolevalue);

&availability = &user.createElement("availability");

&ret = &availability.addText(&availabilityvalue);

&rate = &user.createElement("rate");

&ret = &rate.addText(&ratevalue);

&relevancy = &user.createElement("relevancy");

&ret = &relevancy.addText(&relevancyvalue);

&rating = &user.createElement("rating");

&ret = &rating.addText(&ratingvalue);
```

```

/* Add a comment */

&ret = &user.addComment("Added a new tag for hobbies");

/* Add another tag, hobbies, to the user tag */

&hobbies = &user.createElement("hobbies");

&ret = &hobbies.addText(&hobbiesvalue);

End-While;

&GetCandidatesSQL.Close();

/* convert the BiDoc to an XML string*/

&xmlString = &rootDoc.GenXMLString();

/* Send the Response Object */

%Response.Write(&xmlString);

End-Function;

```

Listing of the XML Request

The PeopleCode program written for the Incoming Business Interlink example expects to receive the following XML request.

```

<?xml version="1.0"?>

<postreq>

  <userid>14682</userid>

  <password>play2win</password>

  <email>joe_blow@peoplesoft.com</email>

  <projtitle></projtitle>

  <projref>PSFT</projref>

  <location>35</location>

```

```

    <country>USA</country>

    <startdate>12-Dec-1999</startdate>

    <duration>lessthan1month</duration>

    <mincomp>45</mincomp>

    <compunit>Hourly</compunit>

    <compcurrency>$</compcurrency>

    <role>25</role>

    <expiredday>30</expiredday>

    <description>testing...</description>

    <skills>JAVA</skills>

</postreq>

```

Listing of the XML Response

The PeopleCode program written for the Incoming Business Interlink example will send the following XML request, assuming that the database contains Joe Blow and Jane Doe, and they both know Java.

```

<?xml version="1.0"?>

<postreqresponse>

  <error>

    <errorcode>1</errorcode>

    <errortext></errortext>

  </error>

  <candidates>

    <user>

      <conid>0000001000</conid>

      <username>Joe Blow</username>

      <comments>Joe Blow comment: Goodbye!</comments>

      <location scenery="great" density="low" blank="eh?">

        <city>San Rafael</city>

        <state>CA</state>

```

```
<zip>94522</zip>

<country>US</country>

</location>

<desiredloc>No Preference</desiredloc>

<role>Database</role>

<availability>2000-10-28</availability>

<rate>0000000045</rate>

<relevancy>10</relevancy>

<rating>0</rating>

<!--Added a new tag for hobbies-->

<hobbies>saying goodbye</hobbies>

</user>

<user>

  <conid>0000001001</conid>

  <username>Jane Doe</username>

  <comments>Jane comment: hello!</comments>

  <location scenery="so-so" density="dense" blank="some stuff">

    <city>Concord</city>

    <state>NJ</state>

    <zip>32444</zip>

    <country>US</country>

  </location>

  <desiredloc>South Bay</desiredloc>

  <role>Developer</role>

  <availability>2000-10-31</availability>

  <rate>0000000100</rate>

  <relevancy>19</relevancy>

  <rating>2</rating>

  <!--Added a new tag for hobbies-->
```

```
<hobbies>saying hello</hobbies>

</user>

</candidates>

</postreqresponse>
```

Test Example: XML Design-Time Plug-in Used with an Incoming Business Interlink

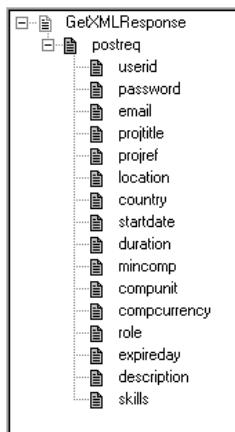
The following XML code is a Business Interlink XML design-time plug-in. This plug-in allows a PeopleCode application to send the XML request that is used as the Incoming Business Interlink example, and then receive the XML response that is used by the Incoming Business Interlink example.

This plug-in consists of a transaction whose inputs are the XML request, and whose outputs are the XML response.

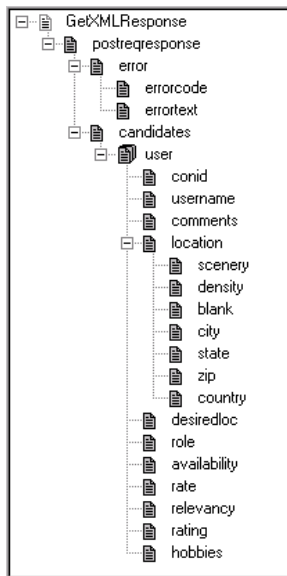


For more information about writing this type of XML design-time plug-in, see PeopleSoft Business Interlink Design-Time Plug-in Programming Guide, Writing a XML Design-Time Plug-In using the pshtpenable Runtime Plug-In.

For reference, here is the structure of this XML design-time plug-in.



Inputs for Transaction: XML Request



Transaction Outputs: XML Response

```

<?xml version="1.0"?>

<interface_driver>

  <general_info>

    <description>Incoming Business Interlink Demo Prototype Plug-
in</description>

    <version>1</version>

    <lastupdate>04/19/00</lastupdate>

    <comments/>

  </general_info>

  <driver_settings>

    <option type="static_catalog" supported="true"/>

    <option type="transaction" supported="true"/>

    <option type="input_class_expandable" supported="true"/>

    <option type="output_class_expandable" supported="true"/>

    <option type="hierarchical_model" supported="true"/>

    <relational_op/>

    <logical_op/>

  </driver_settings>

  <config_parameters>

```

```

<URL>file://pshttpenable.dll</URL>

<parameter name="Method"

    type="enum(GET,POST) " required="true" default="POST"/>

<parameter name="Accept_Type"

    type="enum(text/xml/html) " required="true"

    default="text/xml/html"/>

<parameter name="Content_Type"

    type="enum(Content-Type: text/html,Content-Type: text/xml; charset=utf-
8,Content-Type: application/x-www-form-urlencoded) "

    required="true"

    default="Content-Type: application/x-www-form-urlencoded"/>

<parameter name="MerchantURL" type="string"

    required="true"
default="http://localhost/servlets/xmllink/peoplesoft8/SkillsVillageSearch"/>

<parameter name="InDataType" type="enum(XML,HTML,DHTML) "

    required="true" default="XML"/>

<parameter name="OutDataType" type="enum(XML,HTML,DHTML) "

    required="true" default="XML"/>

<parameter name="XML_Validation" type="bool" required="true"

    default="FALSE"/>

<parameter name="Input_File_Name" type="string"

    required="false" default=""/>

<parameter name="Request_File_Name" type="string"

    required="false" default="c:\temp\DebugQueryString.txt"/>

<parameter name="Simulated_Response_File_Name" type="string"

    required="false" default=""/>

<parameter name="Response_File_Name" type="string"

    required="false" default="c:\temp\DebugResponse.txt"/>

<parameter name="Metatag" type="string" required="false"

    default=""/>

```

```

    <parameter name="userid" type="string" default="PTDMO"
        required="true"/>

    <parameter name="pwd" type="string" default="PTDMO"
        required="true"/>

</config_parameters>

<class_catalog>

    <category name="Incoming Business Interlink Demo">

<!-- The postreq class contains the structure for the input for this
transaction. To see the XML that this produces for the Incoming Business
Interlink, see Listing of the XML Request -->

        <class name="postreq">

            <member name="userid" type="string" default="14682"
                required="false"/>

            <member name="password" type="string" default="play2win"
                required="false"/>

            <member name="email" type="string"
                default="joe_blow@peoplesoft.com"
                required="false"/>

            <member name="projtitle" type="string" default=""
                required="false"/>

            <member name="projref" type="string" default="PSFT"
                required="false"/>

            <member name="location" type="int" default="35"
                required="false"/>

            <member name="country" type="string" default="USA"
                required="false"/>

            <member name="startdate" type="string"
                default="12-Dec-1999" required="false"/>

            <member name="duration"

```

```

        type="enum(lessthan1month, 1to3months, 3to6months,
        6to12months, 12plusmonths)"

        default="lessthan1month" required="false"/>
<member name="mincomp" type="float" default="45"
        required="false"/>
<member name="compunit"
        type="enum =(Daily, Hourly, Monthly, Weekly,Yearly)"
        default="Hourly" required="false"/>
<member name="compcurrency" type="enum =($,?) "
        default="$" required="false"/>
<member name="role" type="int" default="25"
        required="false"/>
<member name="expireday" type="enum(30,60,90) "
        default="30" required="false"/>
<member name="description" type="string"
        default="testing..." required="false"/>
<member name="skills" type="string" default="JAVA"
        required="false"/>
</class>

```

<!--The postreqresponse class contains the structure for the output for this transaction. To see the XML that this produces for the Incoming Business Interlink, see Listing of the XML Response -->

! The postrequestresponse class contains the error class and the
! candidates class.

```

<class name="postreqresponse">
    <member name="error" type="object" classname="error"
        default=""/>
    <member name="candidates" type="object"
        classname="candidates" default=""/>

```

```
</class>

<class name="error">

    <member name="errorcode" type="int" default="1"

        required="false"/>

    <member name="errortext" type="string" default=""

        required="false"/>

</class>

<class name="candidates">

    <member name="user" type="list_object"

        classname="userinfo" default=""/>

</class>

<class name="userinfo">

    <member name="conid" type="string" default=""

        required="false"/>

    <member name="username" type="string" default=""

        required="false"/>

    <member name="comments" type="string" default=""

        required="false"/>

    <member name="location" type="object"

        classname="location" default=""/>

    <member name="desiredloc" type="string" default=""

        required="false"/>

    <member name="role" type="string" default=""

        required="false"/>

    <member name="availability" type="string" default=""

        required="false"/>

    <member name="rate" type="float" default=""

        required="false"/>

    <member name="relevancy" type="int" default=""
```

```

        required="false"/>

        <member name="rating" type="int" default=""

        required="false"/>

        <member name="hobbies" type="string" default=""

        required="false"/>

    </class>

    <class name="location">

        <member name="scenery" type="string"

        attribute="xml_attr" default="" required="false"/>

        <member name="density" type="string"

        attribute="xml_attr" default="" required="false"/>

        <member name="blank" type="string" attribute="xml_attr"

        default="" required="false"/>

        <member name="city" type="string" default=""

        required="false"/>

        <member name="state" type="string" default=""

        required="false"/>

        <member name="zip" type="string" default=""

        required="false"/>

        <member name="country" type="string" default=""

        required="false"/>

    </class>

</category>

</class_catalog>

```

!Here is the transaction containing the input/request postreq class, and the output/response postreqresponse class.

```

<trans_catalog>

```

```
<category name="Incoming Business Interlink Demo">

  <transaction name="GetXMLResponse">

    <input_list>

      <input name="postreq" type="object"

        classname="postreq" default="" required="false"/>

    </input_list>

    <output_list>

      <output name="postreqresponse" type="object"

        classname="postreqresponse" required="true"/>

    </output_list>

  </transaction>

</category>

</trans_catalog>

</interface_driver>
```


CHAPTER 6

Using the Business Interlink Object Methods

This section describes the methods that are associated with the Business Interlink Object. It contains information to help you decide which of these methods you should use, and lists and describes each method.

After you instantiate a Business Interlink Object, you'll use the Business Interlink Object methods to:

- Add input values to the Business Interlink Object.
- Execute the Business Interlink Object.
- Get the output values from the Business Interlink Object.

After the Business Interlink object is instantiated, you can assign values from constants, PeopleSoft variables, or record fields to the inputs of that Business Interlink Object.

When you execute the Business Interlink Object, it loads the appropriate Business Interlink Plug-in and passes itself to that Business Interlink Plug-in. The Business Interlink Plug-in processes the input data, passing the input values of the Business Interlink Object to the external system and then fills the output values of the Business Interlink Object (if there are outputs).

Deciding Which Methods To Use

You'll need to write a PeopleCode program to instantiate the Business Interlink Object from the Business Interlink Definition, using the **GetInterlink** function.

Executing the Business Interlink Object

In most cases, you must use the **Execute** method to execute the Business Interlink object. However, for bulk input, you can use the **BulkExecute** method instead.

The **Execute** and **BulkExecute** methods return a value you can use for status and error checking.

Supporting Standard Input/Output: BIDocs Methods



To better illustrate the BIDocs methods, the examples were created from a modified version of a UPS plug-in. The input parameter `input-param1` and output parameters `output_param1` and `output_param2_list` were added to this example, and the `Account_Info` input parameter was modified to be a list, `Account_Info_List`.

A Business Interlink object can use hierarchical input and output data. The BIDocs methods are the methods that you will most often use to access this data. When you generate a PeopleCode template, these methods are used in that template.

You use BIDocs methods to add input to a Business Interlink object, then you call the `Execute` method, then you use BIDocs methods to get output from the Business Interlink object.

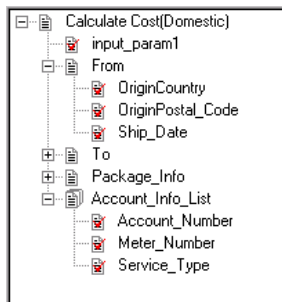


For more information about the BIDocs methods, see [BIDocs Hierarchical Methods](#).

You can think of the Input Doc and the Output Doc that stores the hierarchical data as a tree, with a root doc, node docs, and values.

Input Docs

The following shows an example of an Input Doc:



Example Input Doc

The **root doc** is `Calculate Cost(Domestic)`. A root doc can contain both values and node docs.

The **node docs** are `From`, `To`, `Package_Info` and `Account_Info_List`. Each node can contain both values and child nodes. Node docs can be further described as the following:

- **Simple node docs** have only one set of values for a single instance of a root doc. `From`, `To`, `Package_Info` are all simple node docs.
- **List node docs** can contain more than one set of values for a single instance of a root doc. `Account_Info_List` is a list node doc.

The **values** in the From node are OriginCountry, OriginPostal_Code and Ship_Date. The value within the root node is input_param1. Notice that there are values both within the root doc and within node docs.

Business Interlinks support Input Docs with the following methods:

- GetInputDocs
- AddDoc
- AddValue
- AddNextDoc

The **GetInputDocs** method creates an Input Doc and returns a reference to its root doc. From the above example, it returns a reference to Calculate Cost(Domestic).

Use the **AddDoc** BIDocs method to return a reference to the node docs. From the above example, you would use AddDoc to access the From, To, Package_Info and Account_Info_List node docs. If any of these nodes contain nodes, you use AddDoc to access those as well.

Use the **AddValue** BIDocs method to set values. From the above example, you would use AddValue to set the value for input_param1, and for the From node, to set the values OriginCountry, OriginPostal_Code, and Ship_date. You must call AddDoc on a node before you can call AddValue for its values.

Use the **AddNextDoc** BIDocs method to access the following:

- Use AddNextDoc to return a reference to the next root doc.
- If a node doc is a list, use AddNextDoc to return a reference to the next node doc in the list.

The following code example sets values for the node doc From, which is a simple node doc. It also sets the values for Account_Info_List, which is a list node doc.

```
&Calc_Input = &QE_FEDEX_COST.GetInputDocs("");

&FromDoc = &Calc_Input.AddDoc("From");

&ret = &FromDoc.AddValue("OriginCountry", "United States");

&ret = &FromDoc.AddValue("OriginPostal_Code", &ORIGIN);

&ret = &FromDoc.AddValue("Ship_Date", &SHIPDATE);

&Account_Doc = &Calc_Input.AddDoc("Account_Info_List");

&ret = &Account_Doc.AddValue("Account_Number", "CT-8001");

&ret = &Account_Doc.AddValue("Meter_Number", &METER);

&ret = &Account_Doc.AddValue("Service_Type", &MODE);
```

```

/* add next set of values in list */

&Account_Doc = &Account_Doc.AddNextDoc();

&ret = &Account_Doc.AddValue("Account_Number", "CT-8002");

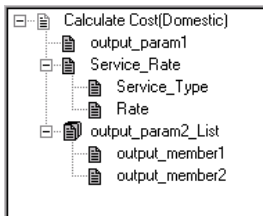
&ret = &Account_Doc.AddValue("Meter_Number", &METER);

&ret = &Account_Doc.AddValue("Service_Type", &MODE);

```

Output Docs

The following shows an example of an Output Doc:



Example Output Structure

The **root doc** is Calculate Cost(Domestic). A root doc can contain both values and node docs.

The **node docs** are Service_Rate and output_param2_List. Each node can contain both values and child nodes. Node docs can be further described as the following:

- **Simple node docs** have only one set of values for a single instance of a root doc. Service_Rate is a simple node doc.
- **List node docs** can contain more than one set of values for a single instance of a root doc. output_param2_List is a list node doc.

The **values** in the Service_Rate node are Service_Type, Rate, and in the output_param2_List node, output_member1 and output_member2, and within the root node, output_param1.

Business Interlinks support hierarchical output structures with the following methods:

- GetOutputDocs
- GetDoc
- GetNextDoc
- GetPreviousDoc
- GetStatus

- GetCount
- MoveToDoc

The **GetOutputDocs** method creates an Output Doc and returns a reference to its root doc. From the above example, it returns a reference to Calculate Cost(Domestic).

Use the **GetDoc** BIDocs method to return a reference to the node docs. From the above example, you would use GetDoc to access the Service_Rate or output_param2_List node docs. If any of these nodes contain nodes, you use GetDoc to access those as well.

Use the **GetValue** BIDocs method to retrieve the values. From the above example, you would use GetValue to retrieve the values for output_param1, and for the Service_Rate node, to get the values Service_Type, Rate, output_member1 and output_member2. You must call GetDoc on a node before you can call GetValue for its values.

Use the **GetNextDoc** and **GetPreviousDoc** BIDocs methods to access the following:

- Use GetNextDoc/GetPreviousDoc to return a reference to the next/previous root doc.
- If a node doc is a list, use GetNextDoc/GetPreviousDoc to return a reference to the next/previous node doc in the list.

Use the **GetCount** method to return either the number of docs in a list node doc, or the number of root docs. In the example, you can count the number of Calculate Cost(Domestic) nodes or the number of output_param2_list nodes.

Use the **MoveToDoc** method to move to either a particular doc at the root level or to a list node doc. In the example, you can move to a Calculate Cost(Domestic) node or to an output_param2_list node.

The following code example gets values for the node docs Service Rate, which is a simple node doc. It also sets the values for output_param2_List, which is a list node doc.

```
&Calc_Input = &QE_FEDEX_COST.GetOutputDocs("");

&Service_Rate_Doc = &Calc_Input.GetDoc("Service_Rate");

&ret = &Service_Rate_Doc.GetValue("Service_Type", "Overnight");

&ret = &Service_Rate_Doc.GetValue("Rate", "50.00");

&Out_Param_Doc = &Calc_Input.GetDoc("output_param2_List");

&ret = &Out_Param_Doc.GetValue("output_member1", "value1");

&ret = &Out_Param_Doc.GetValue("output_member2", "value2");

/* get next set of values in list */
```

```
&Account_Doc = &Out_Param_Doc.AddNextDoc();  
  
&ret = &Out_Param_Doc.GetValue("output_member1", "value3");  
  
&ret = &Out_Param_Doc.GetValue("output_member2", "value4");
```

Supporting Batch Input and Output

The methods discussed in this chapter, except for **BulkExecute**, add input values to the Business Interlink Object one set, or row, at a time. The call to the Business Interlink Plug-in occurs only once. All the input data is passed with the single **Execute**. All output is returned as batch as well. The methods then get the output values one set, or row, at a time.

If you're sending a large amount of data to the input buffers, instead of adding one input row at a time, you might write the data to a staging table, then use the **BulkExecute** method. This method automatically executes; that is, you don't have to use the **Execute** method. It also automatically fills the output record specified with the method with all the output values in every row in the output buffer if you've specified an output record.

Supporting Rowsets

If your data is mapped into rowsets, you may want to use the **InputRowset** method. This method will take a standard rowset object to populate the inputs for the Business Interlink Object. You can use the **FetchIntoRowset** method to repopulate the rowset with output from the Business Interlink object.

Using the Flat Table Input/Output Methods

If your data is in a flat table structure, you can use the flat table methods. **AddInputRow** adds rows of input to the Business Interlink Object; **FetchNextRow** fetches rows of output from the Business Interlink Object.

Supporting Dynamic Output

A Business Interlink can have dynamic output, meaning that the outputs for a Business Interlink object are changeable in data type or number of outputs. (For comparison, a static catalog would mean that the parameters or data members of the Business Interlink are not changeable in type or in number.)

When you have dynamic output, use methods to extract outputs from the Business Interlink Object output buffer. These methods support dynamic output to interrogate the output buffer programmatically to determine the number of fields (columns), their types, and their values.

You can use the **MoveFirst** method to move to the first column, first row of the output buffer, and within a loop, use the **MoveNext** method to move to each row on the output buffer.

Within the **MoveFirst/MoveNext** loop, you can use the **GetFieldCount** method to get the number of columns in the output buffer, which is also the number of outputs for this Business Interlink object. Then you can extract the outputs from the buffer in a loop.

Within the **GetFieldCount** loop, you can use **GetFieldName** to get the name of the output, **GetFieldValue** to get the value of the output (which will be returned as a string), and **GetFieldType** to get the type of the output (if the output is not a string type, you will convert it to this type).

Using the Incoming Business Interlink Methods and Properties

The Incoming Business Interlink methods allow you to parse an XML request and build an XML response.

The **GetContentBody** Request object method converts the request content (an XML request) into an XML string. You can then use this string with the **GetBiDoc** function to create a BiDoc structure that is the same shape and contains the same data as the XML string.

Once you have the BiDocs structure containing the XML request, you can:

- Use the **GetNode** method to get one of the XML elements.
- Use the **NodeType** method to get the type of the node (element, processing instruction, comment).
- Use the **NodeName** property to get the name of the element.
- Use the **NodeValue** property to get the value of the element.

You can also use the standard BiDocs methods (such as **GetDoc**, **GetValue**) to retrieve information from this BiDocs object.

When an XML element contains attributes, the **AttributeCount** property gets the number of attributes, the **GetAttributeName** method gets the name of an attribute, and the **GetAttributeValue** method gets the value of an attribute.

To build an XML response, you can use the **GetBiDocs** function to create a blank BiDocs structure. To create the XML structure within that BiDocs, use **CreateElement** to create an XML element, **AddComment** to add an XML comment, **AddAttribute** to add an attribute to an XML element, and **AddProcessInstruction** to add a processing instruction (the first tag of the XML response). Use the Incoming Business Interlink function **GenXMLString** to create an XML string from the BiDocs structure. You can use the **Write Response** object method to send the string as an XML response.



For more information on the Incoming Business Interlink methods and functions, see Incoming Business Interlink BiDocs Methods And Properties, **GetBiDoc**, and **GetContentBody**.

State of a Business Interlink Object

Your PeopleSoft Business Interlink API should be stateless, that is, if you want to save information from one call of the Business Interlink Object to the next, you will have to do it yourself by writing the relevant information to the database. If you use the **Execute** method more than once within a single PeopleCode event (that is, if you have the **Execute** method in some sort of loop) the state will be preserved. Once you leave the event, any state associated with the Business Interlink Object is lost.

You should only create one Business Interlink Object, that is, you should only use the **GetInterlink** function once. After that, you can load it with data, pass the data to the Business Interlink Plug-in (via **Execute**) and fetch output data as many times as you need.

Declaring a Business Interlink Object

You should declare a Business Interlink object using the data type `Interlink`. In a regular PeopleCode program, you can only declare a Business Interlink object as local. However, in an Application Engine program, you can declare a Business Interlink object as global. Instantiating a Business Interlink object once as a global saves on the significant overhead of reinstantiating a local object for every iteration of PeopleCode called in a loop.

- Global Business Interlink objects can only be used in Application Engine PeopleCode programs because PeopleCode that runs on an application server must be stateless.
- When a restartable Application Engine program abends, global Business Interlink objects that were instantiated before the last checkpoint are automatically reinstantiated at restart. So the object will be available, even though no call has been made to **GetInterlink** in the restarted process. However, the associated Business Interlink data buffers are *not* recovered, so the Application Engine program must be written such that these buffers are empty whenever a checkpoint is taken.
- Business Interlink objects should *not* be declared as global unless they are used in several PeopleCode actions, or in a PeopleCode action that is called in a loop. Only in these instances is the overhead of checkpointing them worthwhile.

Scope of a Business Interlink Object

A Business Interlink object can be instantiated from PeopleCode.

This object can be used anywhere you have PeopleCode, that is, in message subscription PeopleCode, Application Engine PeopleCode, record field PeopleCode, etc.

Business Interlink Object Methods

This section describes the PeopleCode methods you use to instantiate a Business Interlink Object and execute that object. You can instantiate an instance of a named Business Interlink Object using the PeopleCode `GetInterlink` function and then use methods (**AddInputRow**, **Execute**, **FetchNextRow**) on that instance to add data to the object, execute the object, and fetch output data from the object. If any of these methods fail due to error, either an error is displayed and execution of the PeopleCode stops, or the error is logged and processing continues, depending on the setting of the `StopAtError` property.

The BIDocs methods support hierarchical data. Use them if your input and output is in hierarchical form.



For more information about the BIDocs methods, see `AddDoc`.

AddInputRow

Syntax

```
AddInputRow(inputname, value)
```

where *inputname* and *value* are in matched pairs, in the form:

```
inputname1, value1 [, inputname2, value2] . . .
```

Description

The **AddInputRow** method adds a row of input data (*value*) **from** PeopleCode variables or record fields **to** the specified input names (*inputname*) for the Business Interlink Object executing the method. These must be entered in matched pairs, that is, every input name must be followed by its matching value.



The input **name**, not the input path, of the interface definition is used for this method.

There must be an *inputname* for every input parameter defined in the interface definition used to instantiate the Business Interlink Object.

If you specify a record field that is not part of the record the PeopleCode program is associated with, you must use *recname.fieldname* for that *value*.

You can specify default values for every input name in the interface definition (created in the Application Designer.) These values will be used if you generate a PeopleCode template by dragging the interface definition from the Project window in the Application Designer to an open PeopleCode editor window.



For more information on generating a PeopleCode template, see [Generating the PeopleCode Template](#).

Parameters

<i>inputname</i>	Specify the input name. There must be one <i>inputname</i> for every input name defined in the interface definition used to instantiate the Business Interlink Object.
<i>value</i>	Specify the value for the input name. This can be a constant, a variable, or a record field. Each <i>value</i> must be paired with an <i>inputname</i> .

Return Value

A Boolean value: True if the input values were successfully added. Otherwise, it returns False.

Example

In the following example, the Business Interlink Object name is QE_FEDEX_COST, and the input names, such as OriginPostal_Code, are being bound to variables like &ORIGIN, or to literals, like 10 for quantity. The input names could also be bound to record fields.

```
Local Interlink &QE_FEDEX_COST;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

&QE_FEDEX_COST.AddInputRow("OriginCountry", "United States",
    "OriginPostal_Code", &ORIGIN,
    "Ship_Date", &SHIPDATE,
    "DestCountry", "United States",
    "DestPostal_Code", &DESTINATION,
    "Weight", &WEIGHT,
    "Weight_Type", "LBS",
    "Declared_Value", &VALUE,
    "Account_Number", &ACCOUNT,
    "Meter_Number", &METER,
    "Service_Type", &SVCTYPE);
```

Related Topics

Execute, FetchNextRow, AddDoc

BulkExecute

Syntax

```
BulkExecute(RECORD.inputrecname [, RECORD.outputrecname] [, {user_process_inst | user_operid}] )
```

Description

The **BulkExecute** method uses the data in the specified record to populate the input buffer, copying **like-named** fields. Then the method executes, and, optionally, fills the record specified by *outputrecname* with data from the Business Interlink output buffer. **BulkExecute** will result in significantly faster performance for transactions that process large amounts of data. Instead of adding one input row at a time, then fetching the values one at a time, you might write the data to a staging table, use the **BulkExecute** method and then read the data from the output table. This would be especially effective in Application Engine programs that process sets of data rather than individual rows.

This method assumes that the names of the fields in the record match the names of the inputs (or outputs) defined in the Business Interlink Definition. If there is no field in the *inputrecname* for a Business Interlink input parameter, the parameter's default value is used. If no default is specified, an empty string is passed. It's up to the programmer to ensure that this default value is legitimate.

Fields in the output buffer are matched against the fields in the specified in the output record. You do not have to specify an output record. If you don't specify an output record, the output buffers will **not** be populated.



Note. Before you use this method, you should flush the record used for output and remove any residual data that might exist in it.

If you specify an output record, and you want to use the output record for error checking, you must add the following fields to your output record:

- RETURN_STATUS
- RETURN_STATUS_MSG



Note. The field names in the output record must match these names exactly.

Then you must mark one or more input parameters as "key fields" in the Business Interlink definition (using the BulkExecute ID checkbox). The value of these key fields will be copied to

the output record, and you can use these fields to match error messages with input (or output) rows.



For more information about the BulkExec ID checkbox, see the example below and Input Tab: Setting Inputs For Your Business Interlink Definition.

If you want to order your input (and output) rows, you must add the following column to both the input and output table:

- BI_SEQ_NUM



Note. The field name in your input and output records must match this name exactly.

The input rows will be read in the order of numbers in BI_SEQ_NUM, and the output rows will be generated using the same order number.

This method automatically executes, so you don't need to use the **Execute** method with this method.

Whether this method halts on execution or not depends on the setting of the StopAtError configuration parameter. The default value is True, that is, stop if the error number returned is something other than a 1 or 2. This configuration parameter must be set **before** using the **BulkExecute** method.

Parameters

RECORD.*inputrecname*

Specify a record (SQL table) that contains the data you want to use to populate the input buffer of the Business Interlink. If this record has an integer column named BI_SEQ_NUM, that column must contain a sequence number which corresponds to the order in which each row of the SQL table will be read by **BulkExecute**.

RECORD.*outputrecname*

Specify a record (SQL table) that will hold the data that populates the output buffer of the Business Interlink. This is optional; when used, it is often used to error check the input. If *outputrecname* is NONE, **BulkExecute** will not fill an output record. If this record has an integer column named BI_SEQ_NUM, that column contains a sequence number which corresponds to the order in which each row of the SQL table was written by **BulkExecute**; there will be a one-to-one correspondence with the numbers in the BI_SEQ_NUM column in **RECORD**.*inputrecname*.

user_process_inst |
user_operid

This is an optional parameter that allows either different Application Engine programs or different clients to populate the same **RECORD.outputrecname** at the same time

For *user_process_inst*, the parameter takes an integer. **RECORD.outputrecname** must have a **PROCESS_INSTANCE** field. The **PROCESS_INSTANCE** field is used to identify the Application Engine program that is using this Business Interlink. You can use the **%PROCESS_INSTANCE** variable to populate *user_process_inst*.

For *user_operid*, the parameter takes a string. **RECORD.outputrecname** must have an **OPERID** field. The **OPERID** field is used to identify the client who is using this Business Interlink. You can use the **%OPERID** variable to populate *user_operid*.

Returns

Number	Meaning
1	The Business Interlink Object executes successfully.
2	The Business Interlink Object failed to execute.
3	Transaction failed
4	Query failed
5	Missing criteria
6	Input mismatch
7	Output mismatch
8	No response from server
9	Missing parameter
10	Invalid username
11	Invalid password
12	Invalid server name
13	Connection error
14	Connection refused
15	Timeout reached
16	Unequal lists
17	No data for output
18	Output parameters empty
19	Driver not found

20	Internet connect error
21	XML parser error
22	XML deserialize
201	Warning: Field too large (execute succeed)
202	Ignore sign number (execute succeed)
203	Convert Float to integer (execute succeed)

Example

Here are two PeopleSoft records that could be used as input and output records for BulkExecute. The key fields in the input record, which need to have the BulkExecute ID checkbox checked in the Application Designer, are QE_RP_PO_NUMBER and QE_RP_SITENAME. These key fields are used both for input and output.

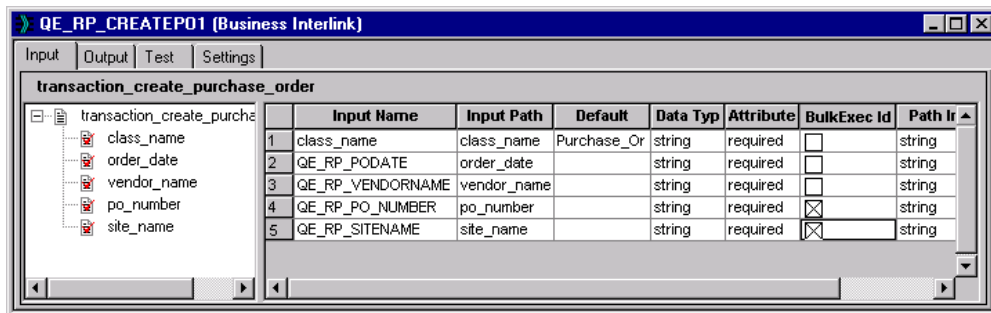
Num	Field Name	Type	Len	Format	H	Short Name	Long Name
1	QE_RP_PO_NUMBER	Char	40	Mixed		QE_RP_PONUMBER	QE_RP_PONUMBER
2	QE_RP_PODATE	DtTm	26			QE_RP_PODATE	QE_RP_PODATE
3	QE_RP_SITENAME	Char	15	Mixed		SITE_NAME	SITE_NAME
4	QE_RP_VENDORNAME	Char	30	Mixed		QE_RP_VENDORNAM	QE_RP_VENDORNAME

Example Input Record for BulkExecute

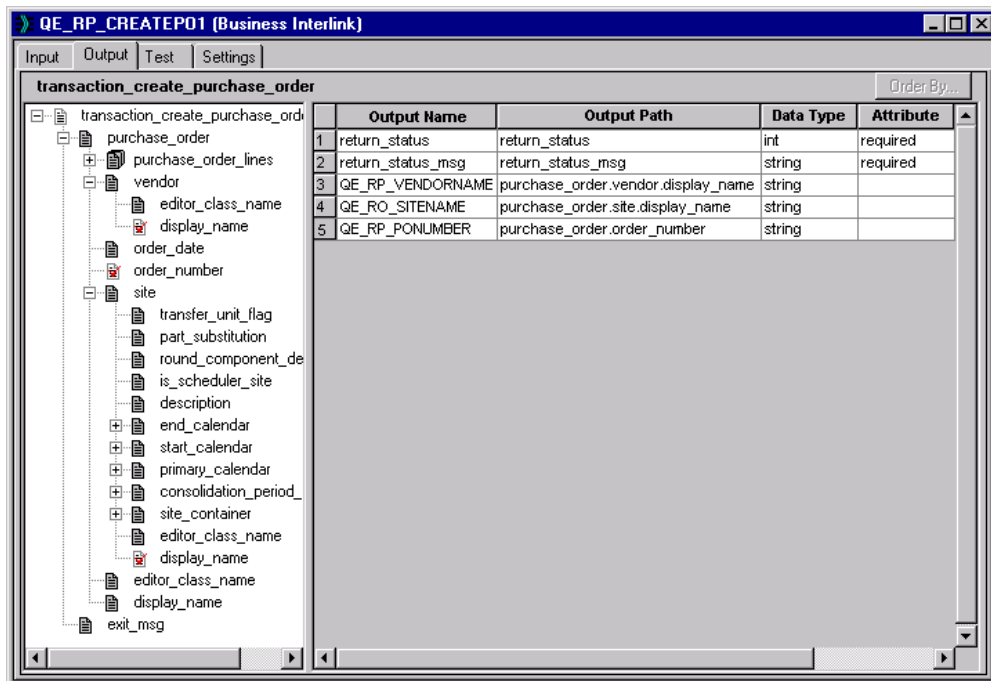
Num	Field Name	Type	Len	Format	H	Short Name	Long Name
1	QE_RP_PO_NUMBER	Char	40	Mixed		QE_RP_PONUMBER	QE_RP_PONUMBER
2	QE_RP_SITENAME	Char	15	Mixed		SITE_NAME	SITE_NAME
3	RETURN_STATUS	Sign	3			RETURN_STATUS	RETURN_STATUS
4	RETURN_STATUS_MSG	Char	254	Mixed		RETURN_STAT_MSG	RETURN_STATUS_MSG

Example Output Record for BulkExecute

Here are the corresponding inputs and outputs for a Business Interlink that has had its inputs and outputs renamed to match the field names in the records. The inputs and outputs have been renamed where necessary to match the record field names.



Example Business Interlink Inputs for BulkExecute



Example Business Interlink Outputs for BulkExecute

The PeopleCode program using these records could contain the following code to run **BulkExecute**.

```
Local Interlink &CREATE_PO;

Local number &EXECSRSLT;

&CREATE_PO = GetInterlink(INTERLINK.QE_RP_CREATEP01);

/* The next three lines set configuration parameters, which are not shown in the
previous examples. */

&CREATE_PO.SERVER_NAME = "bc1";
```

```
&CREATE_PO.RSERVER_HOST = "4.3.4.3";
```

```
&CREATE_PO.RSERVER_PORT = "2200";
```

```
&EXECRSLT = &CREATE_PO.BulkExecute(RECORD.QE_RP_PO, RECORD.QE_RP_PO_OUT1);
```

Related Topics

None

Clear

Syntax

```
Clear()
```

Description

The **Clear** method clears the input and output buffers. If you're using Business Interlinks in a batch program, every time after you use the **Execute** method, you want to use the **Clear** method to flush out the input and output buffers.

Parameters

None.

Returns

None.

Example

```
For &n = 1 to x

    &myinterlink.AddInputRow("input1", value1, "input2", value2);

    &myinterlink.Execute();

    &myinterlink.FetchNextRow("output1", value1, "output2", value2);

    /* do processing on data */

    &myinterlink.Clear();

    &n = &n + 1;

End-for;
```


Related Topics

BulkExecute, Execute

Execute

Syntax

```
Execute ()
```

Description

When a Business Interlink Object executes the **Execute** method, the transaction associated with the Business Interlink Object is executed.

If there is only one row, after appropriate substitution is made, the transaction is executed only once. Otherwise, the data is “batched up” and sent once. You only have to call **Execute** once to execute all the rows of the input buffer.

Generally you would only use the **Execute** method after using the **AddInputRow** or **InputRowset** methods. If you generate a PeopleCode template by dragging the Business Interlink Definition from the Project window in the Application Designer to an open PeopleCode editor window, the usual order of the execution of methods is shown in the template.



For more information on generating a PeopleCode template, see Generating the PeopleCode Template.

If you're using Business Interlinks in a batch program, every time after you use the **Execute** method, you want to use the **Clear** method to flush out the input and output buffers.

Whether this method halts on execution or not depends on the setting of the StopAtError configuration parameter. The default value is True, that is, stop if the error number returned is something other than a 1 or 2. This configuration parameter must be set **before** using the **Execute** method.

Parameters

None.

Returns

Number	Meaning
1	The Business Interlink Object executes successfully.
2	The Business Interlink Object failed to execute.
3	Transaction failed
4	Query failed

5	Missing criteria
6	Input mismatch
7	Output mismatch
8	No response from server
9	Missing parameter
10	Invalid username
11	Invalid password
12	Invalid server name
13	Connection error
14	Connection refused
15	Timeout reached
16	Unequal lists
17	No data for output
18	Output parameters empty
19	Driver not found
20	Internet connect error
21	XML parser error
22	XML deserialize

Example

```

Local number &EXECRESLT;

. . .

&EXECRESLT = &SRA_ALL_1.Execute();

If (&EXECRESLT <> 1) Then

    /* The instance failed to execute */

    /* Do Error Processing */

End-If;
```

Related Topics

AddInputRow, FetchIntoRecord, FetchIntoRowset, GetFieldCount, GetFieldType, GetFieldValue, GetInputDocs, MoveFirst, MoveNext

FetchIntoRecord

Syntax

```
FetchIntoRecord(RECORD.recname [, {user_process_inst | user_operid}])
```

Description

The **FetchIntoRecord** method copies the data from the output buffer into the specified record (SQL table), copying **like-named** fields. This method assumes that the names of the fields in the record match the names of the outputs defined in the Business Interlink Definition.

You can use the **FetchIntoRecord** method to perform a transaction on a large amount of data that you want to receive from your system. Instead of executing your Business Interlink and then fetching one output row at a time, you might execute your Business Interlink, then use the **FetchIntoRecord** method to write the data returned from the Business Interlink to a staging table.



Note. Before you use this method, you should flush the record used for output and remove any residual data that might exist in it.

If your data is hierarchical, i.e., in a rowset, and you want to preserve the hierarchy, use **FetchIntoRowset** instead of this method.

If you want to order your output rows, you must add the following column to the record:

- BI_SEQ_NUM

This column will be populated with a sequence number that corresponds to the order in which each row of the record was written to by the method.

Parameters

RECORD.recname

Specify a record (SQL table) that you want to populate with data from the output buffers. If this record has an integer column named BI_SEQ_NUM, that column contains a sequence number which corresponds to the order in which each row of the SQL table was written by **FetchIntoRecord**.

*user_process_inst |
user_operid*

This is an optional parameter that allows **either** different Application Engine programs **or** different clients to populate the same **RECORD.outputrecname** at the same time.

For *user_process_inst*, the parameter takes an integer. **RECORD.outputrecname** must have a **PROCESS_INSTANCE** field. The **PROCESS_INSTANCE** field is used to identify the Application Engine program that is using this Business Interlink. You can use the **%PROCESS_INSTANCE** variable to populate *user_process_inst*.

For *user_operid*, the parameter takes a string. **RECORD.outputrecname** must have an **OPERID** field. The **OPERID** field is used to identify the client who is using this Business Interlink. You can use the **%OPERID** variable to populate *user_operid*.

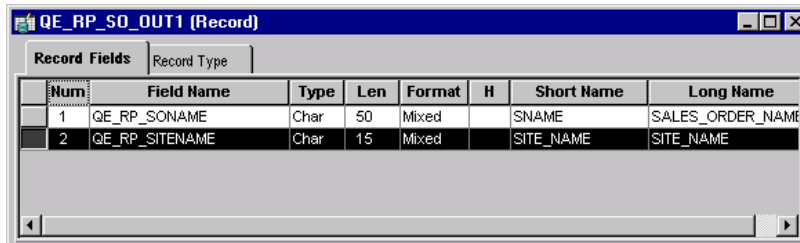
Returns

Number of rows fetched if one or more non-empty rows are returned. If an empty row is returned, that is, every if field is empty except the *return_status* and *return_status_msg* fields, this method returns one of the following error messages:

Number	Meaning
1	NoInterfaceObject
2	ParamCountError
3	IncorrectParameterType
4	NoColumnForOutputParm
5	NoColumnForInputParm
6	BulkInsertStartFailed
7	BulkInsertStopFailed
8	CouldNotCreateSelectCursor
9	CouldNotCreateInsertCursor
10	CouldNotDestroySelectCursor
11	CouldNotDestroyInsertCursor
12	InputRecordDoesNotExist
13	OutputRecordDoesNotExist
14	ContainEmptyRow
15	SQLExecError
201	FieldTooLarge
202	IgnoreSignNumber
203	ConvertFloatToInt

Example

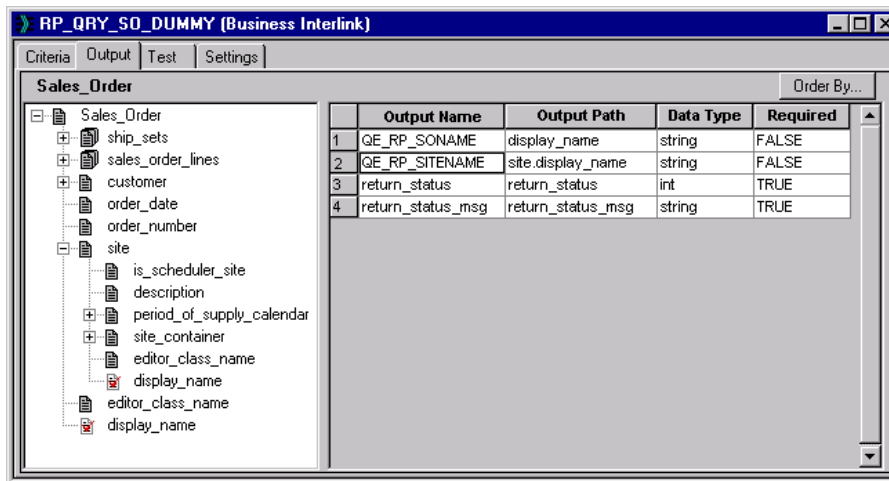
Here is a PeopleSoft record that could be used as an output record for `FetchIntoRecord`.



Num	Field Name	Type	Len	Format	H	Short Name	Long Name
1	QE_RP_SONAME	Char	50	Mixed		SNAME	SALES_ORDER_NAME
2	QE_RP_SITENAME	Char	15	Mixed		SITE_NAME	SITE_NAME

Example Output Record for `FetchIntoRecord`

Here are the corresponding outputs for a Business Interlink that has had its outputs renamed to match the field names in the records.



	Output Name	Output Path	Data Type	Required
1	QE_RP_SONAME	display_name	string	FALSE
2	QE_RP_SITENAME	site.display_name	string	FALSE
3	return_status	return_status	int	TRUE
4	return_status_msg	return_status_msg	string	TRUE

Example Business Interlink Outputs for `FetchIntoRecord`

The PeopleCode program using these records could contain the following code to run **`FetchIntoRecord`**.

```
Local Interlink &RP_QRY_SO_EXAMPLE_1;

Local number &RSLT;

&RP_QRY_SO_EXAMPLE_1 = GetInterlink (INTERLINK.RP_QRY_SO_EXAMPLE);

/* ==> Add Criteria inputs (RHS): */

&RP_QRY_SO_EXAMPLE_1.AddInputRow("IN1_customer.display_name", "Joe Blow" ,
"IN2_order_number", "199" , "IN3_order_date", "09/18/99" , "IN4_order_date",
"10/18/96" );
```

```
&EXECRSLT = &RP_QRY_SO_EXAMPLE_1.Execute();

If ( &EXECRSLT <> 1 ) Then

    /* The instance failed to execute */

Else

    &RSLT =

        &RP_QRY_SO_EXAMPLE_1.FetchIntoRecord(RECORD.QE_RP_SO_OUT1);

End-If; /* If NOT &RSLT ... */
```

Related Topics

AddInputRow, Execute

FetchIntoRowset

Syntax

```
FetchIntoRowset (&Rowset)
```

Description

The **FetchIntoRowset** method copies the data from the output buffer into the specified rowset, copying **like-named** fields. This method assumes that the names of the fields in the rowset match the names of the outputs defined in the Business Interlink definition, **and** that the structure is the same. You can use the **FetchIntoRowset** method to repopulate the rowset with output from the Business Interlink object.



Note. Before you use this method, you should flush the rowset used for output and remove any residual data that might exist in it.

Use this method only if you have a hierarchical data structure and you want to preserve the hierarchy. Otherwise, use **BulkExecute** or **FetchIntoRecord**.

Parameters

&Rowset

Specify rowset object that you want to populate with data from the output buffers. This must be an existing, instantiated rowset.

Returns

Number of rows fetched if one or more non-empty rows are returned. If an empty row is returned, that is, every if field is empty except the return_status and return_status_msg fields, this method returns one of the following error messages:

Number	Meaning
1	NoInterfaceObject
2	ParamCountError
3	IncorrectParameterType
4	NoColumnForOutputParm
5	NoColumnForInputParm
6	BulkInsertStartFailed
7	BulkInsertStopFailed
8	CouldNotCreateSelectCursor
9	CouldNotCreateInsertCursor
10	CouldNotDestroySelectCursor
11	CouldNotDestroyInsertCursor
12	InputRecordDoesNotExist
13	OutputRecordDoesNotExist
14	ContainEmptyRow
15	SQLExecError
201	FieldTooLarge
202	IgnoreSignNumber
203	ConvertFloatToInt

Example

The example uses the rowset on level 1 from the EMPLOYEE_CHECKLIST page. A PeopleCode event running on level 0 in that page can access the child rowset (level 1), shown below from Scroll Bar 1 to Scroll Bar 2.

EMPLOYEE_CHECKLIST.ENG (Panel)								
Panel Designer		Order						
	Lvl	Label	Type	Field	Record	Display Control	Related Display	Related Control
1	0	Frame	Frame			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	0	Frame	Frame			<input type="checkbox"/>	<input type="checkbox"/>	
3	0	Frame	Frame			<input type="checkbox"/>	<input type="checkbox"/>	
4	0	Employee Name	Edit Box	NAME	PERSONAL_DATA	<input type="checkbox"/>	<input type="checkbox"/>	
5	0	ID	Edit Box	EMPLID	PERSONAL_DATA	<input type="checkbox"/>	<input type="checkbox"/>	
6	1	Checklist Item Tbl	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	1	Checklist Sequen	Edit Box	CHECKLIST_SEQ	CHECKLIST_ITEM	<input type="checkbox"/>	<input type="checkbox"/>	
8	1	Scroll Bar 1	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	1	Checklist Date	Edit Box	CHECKLIST_DT	EMPL_CHECKLIST	<input type="checkbox"/>	<input type="checkbox"/>	
10	1	derived_hr.effdt	Edit Box	EFFDT	DERIVED_HR	<input type="checkbox"/>	<input type="checkbox"/>	
11	1	Checklist	Edit Box	CHECKLIST_CD	EMPL_CHECKLIST	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	1	Checklist Descript	Edit Box	DESCR	CHECKLIST_TBL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	11
13	1	Responsible ID	Edit Box	RESPONSIBLE_ID	EMPL_CHECKLIST	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	1	Responsible Nam	Edit Box	NAME	PERSONAL_DATA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	13
15	1	Comment	Long Edit Box	COMMENTS	EMPL_CHECKLIST	<input type="checkbox"/>	<input type="checkbox"/>	
16	2	Scroll Bar 2	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
17	2	Chklist Seq	Edit Box	CHECKLIST_SEQ	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	
18	2	Chklist Itm	Edit Box	CHKLIST_ITEM_CD	EMPL_CHKLIST_ITM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	2	Briefing Descripti	Edit Box	DESCR	CHKLIST_ITEM_TBL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	18
20	2	Briefing Status	Drop Down List	BRIEFING_STATUS	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	
21	2	Status Date	Edit Box	STATUS_DT	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	

EMPLOYEE_CHECKLIST Page Order

EMPL_CHECKLIST is the primary database record for that scrollbar on the EMPLOYEE_CHECKLIST page. The following PeopleCode access the level 1 rowset using EMPL_CHECKLIST. The Business Interlink Object name is QE_BI_EMPL_CHECKLIST. This Business Interlink Object uses the level 1 rowset as its input and its output.

The **InputRowset** method uses this rowset as input for QE_BI_EMPL_CHECKLIST. Then a blank duplicate of the rowset is created with **CreateRowset**, and then the output of QE_BI_EMPL_CHECKLIST is fetched into the blank rowset with **FetchIntoRowset**.

```

&MYROWSET = GetRowset (SCROLL.EMPL_CHECKLIST) ;

&ROWCOUNT = &QE_BI_EMPL_CHECKLIST.InputRowset (&MYROWSET) ;

&RSLT = &QE_BI_EMPL_CHECKLIST.Execute() ;

/* do some error processing */

&WorkRowset = CreateRowset (&MYROWSET) ;

&ROWCOUNT = &QE_BI_EMPL_CHECKLIST.FetchIntoRowset (&WorkRowset) ;

If &ROWCOUNT = 0 Then

    /* do some error processing */

Else

    /* Process the rowset from QE_BI_EMPL_CHECKLIST. Check it for errors. */

    For &I = 1 to &WorkRowset.RowCount

```



```

For &K = 1 to &WorkRowset(&I).RecordCount

    &REC = &WorkRowset(&I).GetRecord(&K);

    &REC.ExecuteEdits();

    For &M = 1 to &REC.FieldCount

        If &REC.GetField(&M).EditError Then

            /* there are errors */

            /* do other processing */

            End-If;

        End-For;

    End-For;

End-for;

End-if;

```

Related Topics

Execute, GetInputDocs, Data Buffer Access, Rowset Class

FetchNextRow

Syntax

```
FetchNextRow(outputname, value)
```

where *outputname* and *value* are in matched pairs, in the form:

```
outputname1, value1 [, outputname2, value2] . . .
```

Description

After the Business Interlink Object executes the method **Execute**, the **FetchNextRow** method can be used to retrieve a row of output and store the values of the output name (*outputname*) to PeopleCode variables or record fields (*value*). These must be entered in matched pairs, that is, every output name must be followed by its matching value.



The output **name**, not the output path of the interface definition, is used for this method.

There must be an *outputname* for every output name defined in the interface definition used to instantiate the Business Interlink Object.

If you specify a record field that is not part of for the record the PeopleCode program is associated with, you must use *recname.fieldname* for that *value*.

Parameters

<i>outputname</i>	Specify the output name. There must be one <i>outputname</i> for every output name defined in the interface definition used to instantiate the Business Interlink Object.
<i>value</i>	Specify the value for the output name. This can be a variable or a record field. Each <i>value</i> must be paired with an <i>outputname</i> .

Return Value

A Boolean value: True if the row of output parameters was fetched. Otherwise, it returns False.

Example

In the following example, the Business Interlink Object name is QE_FEDEX_COST, and the output names, such as **Service_Type**, are being bound to variables, such as &SVCTYPESTR.

```
&RSLT = &QE_FEDEX_COST.FetchNextRow(
    "Service_Type", &SVCTYPESTR,
    "Rate", &RATESTR,
    "return_status", &STATUSSTR,
    "return_status_msg", &MSGSTR);
```

Related Topics

AddInputRow, Execute

GetFieldCount

Syntax

```
GetFieldCount()
```

Description

The **GetFieldCount** method is used to support dynamic output. It returns the number of columns in the output buffer, which is the same as the number of outputs in each row of the output buffer.



This method can also be used with BIDocs data.

Parameters

None.

Returns

Number. The number of columns in the output buffer, which is the same as the number of outputs in each row of the output buffer.

Example

The following example is an edited version of PeopleCode used for a Federal Express Business Interlink named QE_FEDEX_COST_EX that calculates shipping costs. The **MoveNext** method moves to a row in the output buffer; in this case, there is only one row of output. Then the **GetFieldCount** method gets the number of fields, or outputs, in the row, so the **For** loop can process the outputs.

```
Local Interlink &QE_FEDEX_COST;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

/* Set &ORIGIN, &SHIPDATE, &DESTINATION, &WEIGHT, &VALUE, &ACCOUNT, &METER,
&SVCTYPE */

/* Add inputs. This adds one row of data. You would use a loop here if you
wanted to add several rows of data. */

&QE_FEDEX_COST.AddInputRow("OriginCountry", "United States",
"OriginPostal_Code", &ORIGIN, "Ship_Date", &SHIPDATE, "DestCountry", "United
States", "DestPostal_Code", &DESTINATION, "Weight", &WEIGHT, "Weight_Type",
"LBS", "Declared_Value", &VALUE, "Account_Number", &ACCOUNT, "Meter_Number",
&METER, "Service_Type", &SVCTYPE);

&EXECRESLT = &QE_FEDEX_COST.Execute();

If (&EXECRESLT <> 1) Then

    /* The instance failed to execute */

Else

    &FIELDcount = &QE_FEDEX_COST.GetFieldCount();

    &OUTSTRING = "FieldCount : " | &FIELDcount | &CRLF;

    &RSLR = &QE_FEDEX_COST.MoveNext();

    For &I = 0 To &FIELDcount - 1
```

```

&NAME1 = &QE_FEDEX_COST.GetFieldName(&I);

&VALUE1 = &QE_FEDEX_COST.GetFieldValue(&I) | &CRLF;

&TEMPSTRING = &OUTSTRING;

&OUTSTRING = &TEMPSTRING | &NAME1 | ":" | " " | &VALUE1

End-For;

GetLevel0().GetRow(1).GetRecord(Record.QE_FEDEX_WRK).

QE_FEDEX_RESULTS.Value = &OUTSTRING;

End-If

```

Related Topics

AddInputRow, Execute, GetFieldType, GetFieldValue, MoveFirst, MoveNext, Supporting Standard Input/Output: BIDocs Methods

GetFieldType

Syntax

```
GetFieldType(index)
```

Description

The **GetFieldType** method is used to support dynamic output. It returns the type of field specified by *index*. You can only use this method after you have used the **MoveFirst/MoveNext** methods: otherwise the system doesn't know where to start.



This method can also be used with BIDocs data.

Parameter

<i>index</i>	Specify the number of the field you want to find the type of.
--------------	---

Returns

A number indicating the type of the field. The valid values are:

Value	Description
-1	error
1	String

2	Integer
3	Float
4	Boolean
5	Date
6	Time
7	DateTime
8	Binary
9	Object

Example

In the following example, the Business Interlink Object name is &MYBI. The example uses **MoveFirst** to move to the first row of the output buffer, and to the first column, or first field, in that row. The **Repeat** loop uses **MoveNext** to go through every row in the output buffer. The **For** loop processes every field in every row, using the **GetFieldCount** method to get the number of fields, or outputs, in the row. Within the **For** loop, **GetFieldType** gets the type of the field data (string, integer, etc.).

```
/* Add inputs to the Business Interlink Object, then call Execute to execute the
Business Interlink Object. You are then ready to get the outputs using the
following code. */
```

```
If (&MYBI.MoveFirst()) Then

    Repeat

        For &I = 1 to &MYBI.GetFieldCount

            &TYPE = &MYBI.GetFieldType(&I);

            Evaluate &TYPE

            Where = 1

            &STRING_VARIABLE = &MYBI.GetFieldValue(&I);

            /* test for and process other field types */

            End-Evaluate;

        End-For;

    Until Not (&MYBI.MoveNext());

Else

    /* Process error - no output buffer */

End-If;
```

Related Topics

AddInputRow, Execute, GetFieldCount, GetFieldValue, MoveFirst, MoveNext, Supporting Standard Input/Output: BIDocs Methods

GetFieldValue

Syntax

```
GetFieldValue (index)
```

Description

The **GetFieldValue** method is used to support dynamic output. It returns the value of the field specified by *index*. You can only use this method after you have used the **MoveFirst** method: otherwise the system doesn't know where to start.



This method can also be used with BIDocs data.

Parameter

<i>index</i>	Specify the number of the field you want to find the value of.
--------------	--

Returns

A string containing the value of the field.

Example

In the following example, the Business Interlink Object name is &MYBI. The example uses **MoveFirst** to move to the first row of the output buffer, and to the first column, or first field, in that row. The **Repeat** loop uses **MoveNext** to go through every row in the output buffer. The **For** loop processes every field in every row, using the **GetFieldCount** method to get the number of fields, or outputs, in the row. Within the **For** loop, **GetFieldValue** gets the value of the field data.

```
/* Add inputs to the Business Interlink Object, then call Execute to execute the
Business Interlink Object. You are then ready to get the outputs using the
following code. */
```

```
If (&MYBI.MoveFirst()) Then
  Repeat
    For &I = 1 to &MYBI.GetFieldCount
      &TYPE = &MYBI.GetFieldType(&I);
```

```

        Evaluate &TYPE

        Where = 1

        &STRING_VARIABLE = &MYBI.GetFieldValue(&I);

        /* test for and process other field types */

        End-Evaluate;

    End-For;

    Until Not (&MYBI.MoveNext());

Else

    /* Process error - no output buffer */

End-If;

```

Related Topics

AddInputRow, Execute, GetFieldCount, GetFieldType, MoveFirst, MoveNext, Supporting Standard Input/Output: BIDocs Methods

GetInputDocs

Syntax

```
GetInputDocs (" ")
```

Description

The **GetInputDocs** method gets the top BIDocs input document at the root level for a Business Interlink Object. This is a hierarchical structure that will contain the values for the inputs for this Business Interlink Object. The methods that you use to put the input values into this document are the BIDocs methods.



For more information about the BIDocs methods, see BIDocs Hierarchical Methods.

Parameters

None.

Return Value

A BIDocs input document. This is the document at the top of the root level of the BIDocs input document for a Business Interlink Object.

Example

```
Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostOut, &CalcCostIn;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

&CalcCostIn = &QE_FEDEX_COST.GetInputDocs("");

/* You can now insert the input values and execute the Business Interlink
object. */
```

GetOutputDocs

Syntax

```
GetOutputDocs("")
```

Description

The **GetOutputDocs** method gets the top BIDocs output document at the root level for a Business Interlink Object. This is a hierarchical structure that contains the values for the outputs for this Business Interlink Object. The methods that you use to get output values from this document are the BIDocs methods.



For more information about the BIDocs methods, see BIDocs Hierarchical Methods.

Parameters

None.

Return Value

A BIDocs output document. This is the document at the top of the root level of the BIDocs output document for a Business Interlink Object.

Example

```
Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostIn, &CalcCostOut;
```



```

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

/* You can now execute the Business Interlink object and get the output values.
*/

```

InputRowset

Syntax

```
InputRowset (&Rowset)
```

Description

The **InputRowset** method uses the data in the specified rowset to populate the input buffer, copying **like-named** fields in the appropriate structure. This method assumes that the names of the fields in the rowset match the names of the inputs defined in the Business Interlink Definition, **and** that the structure is the same.

Use this method only if you have a hierarchical data structure and you want to preserve the hierarchy. Otherwise, use **BulkExecute** or **FetchIntoRecord**.

Parameters

&Rowset Specify an existing, instantiated rowset object.

Returns

An optional value: the number of rows inserted into the output buffer.

If an empty row is returned, that is, every if field is empty except the `return_status` and `return_status_msg` fields, this method returns one of the following error messages:

Number	Meaning
1	NoInterfaceObject
2	ParamCountError
3	IncorrectParameterType
4	NoColumnForOutputParm
5	NoColumnForInputParm
6	BulkInsertStartFailed
7	BulkInsertStopFailed

8	CouldNotCreateSelectCursor
9	CouldNotCreateInsertCursor
10	CouldNotDestroySelectCursor
11	CouldNotDestroyInsertCursor
12	InputRecordDoesNotExist
13	OutputRecordDoesNotExist
14	ContainEmptyRow
15	SQLExecError

Example

The example uses the rowset on level 1 from the EMPLOYEE_CHECKLIST page. A PeopleCode event running on level 0 in that page can access the child rowset (level 1), shown below from Scroll Bar 1 to Scroll Bar 2.

	Lvl	Label	Type	Field	Record	Display Control	Related Display	Related Control
1	0	Frame	Frame			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	0	Frame	Frame			<input type="checkbox"/>	<input type="checkbox"/>	
3	0	Frame	Frame			<input type="checkbox"/>	<input type="checkbox"/>	
4	0	Employee Name	Edit Box	NAME	PERSONAL_DATA	<input type="checkbox"/>	<input type="checkbox"/>	
5	0	ID	Edit Box	EMPLID	PERSONAL_DATA	<input type="checkbox"/>	<input type="checkbox"/>	
6	1	Checklist Item Tbl	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	1	Checklist Sequen	Edit Box	CHECKLIST_SEQ	CHECKLIST_ITEM	<input type="checkbox"/>	<input type="checkbox"/>	
8	1	Scroll Bar 1	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	1	Checklist Date	Edit Box	CHECKLIST_DT	EMPL_CHECKLIST	<input type="checkbox"/>	<input type="checkbox"/>	
10	1	derived_hr_effdt	Edit Box	EFFDT	DERIVED_HR	<input type="checkbox"/>	<input type="checkbox"/>	
11	1	Checklist	Edit Box	CHECKLIST_CD	EMPL_CHECKLIST	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
12	1	Checklist Descript	Edit Box	DESCR	CHECKLIST_TBL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	11
13	1	Responsible ID	Edit Box	RESPONSIBLE_ID	EMPL_CHECKLIST	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
14	1	Responsible Nam	Edit Box	NAME	PERSONAL_DATA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	13
15	1	Comment	Long Edit Box	COMMENTS	EMPL_CHECKLIST	<input type="checkbox"/>	<input type="checkbox"/>	
16	2	Scroll Bar 2	Scroll Bar			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
17	2	Chklist Seq	Edit Box	CHECKLIST_SEQ	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	
18	2	Chklist Itm	Edit Box	CHKLIST_ITEM_CD	EMPL_CHKLIST_ITM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
19	2	Briefing Descripti	Edit Box	DESCR	CHKLIST_ITEM_TBL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	18
20	2	Briefing Status	Drop Down List	BRIEFING_STATUS	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	
21	2	Status Date	Edit Box	STATUS_DT	EMPL_CHKLIST_ITM	<input type="checkbox"/>	<input type="checkbox"/>	

EMPLOYEE_CHECKLIST Page Order

EMPL_CHECKLIST is the primary database record for that scrollbar on the EMPLOYEE_CHECKLIST page. The following PeopleCode access the level 1 rowset using EMPL_CHECKLIST. The Business Interlink Object name is QE_BI_EMPL_CHECKLIST. This Business Interlink Object uses the level 1 rowset as its input and its output.

The **InputRowset** method uses this rowset as input for QE_BI_EMPL_CHECKLIST. Then a blank duplicate of the rowset is created with CreateRowset, and then the output of QE_BI_EMPL_CHECKLIST is fetched into the blank rowset with **FetchIntoRowset**.

```
&MYROWSET = GetRowset (SCROLL.EMPL_CHECKLIST) ;
```

```

&ROWCOUNT = &QE_BI_EMPL_CHECKLIST.InputRowset(&MYROWSET);

&RSLT = &QE_BI_EMPL_CHECKLIST.Execute();

/* do some error processing */

&WorkRowset = CreateRowset(&MYROWSET);

&ROWCOUNT = &QE_BI_EMPL_CHECKLIST.FetchIntoRowset(&WorkRowset);

If &ROWCOUNT = 0 Then

    /* do some error processing */

Else

    /* Process the rowset from QE_BI_EMPL_CHECKLIST. Check it for errors. */

    For &I = 1 to &WorkRowset.RowCount

        For &K = 1 to &WorkRowset(&I).RecordCount

            &REC = &WorkRowset(&I).GetRecord(&K);

            &REC.ExecuteEdits();

            For &M = 1 to &REC.FieldCount

                If &REC.GetField(&M).EditError Then

                    /* there are errors */

                    /* do other processing */

                    End-If;

                End-For;

            End-For;

        End-For;

    End-for;

End-if;

```

Related Topics

Execute, GetInputDocs, Data Buffer Access, Rowset Class

MoveFirst

Syntax

```
MoveFirst()
```

Description

The **MoveFirst** method is used to support dynamic output. This method places you in the first column, first row of the output buffer. You must use this method **before** you try to access any data.

Parameters

None.

Returns

Boolean: True if successfully positioned cursor at the first column first row of the output buffer, False otherwise.

Example

In the following example, the Business Interlink Object name is &MYBI. The example uses **MoveFirst** to move to the first row of the output buffer, and to the first column, or first field, in that row. The **Repeat** loop uses **MoveNext** to go through every row in the output buffer. The **For** loop processes every field in every row, using the **GetFieldCount** method to get the number of fields, or outputs, in the row.

```
/* Add inputs to the Business Interlink Object, then call Execute to execute the
Business Interlink Object. You are then ready to get the outputs using the
following code. */
```

```
If (&MYBI.MoveFirst()) Then

  Repeat

    For &I = 1 to &MYBI.GetFieldCount

      &TYPE = &MYBI.GetFieldType(&I);

      Evaluate &TYPE

      Where = 1

      &STRING_VARIABLE = &MYBI.GetFieldValue(&I);

      /* test for and process other field types */

      End-Evaluate;

    End-For;

  Until Not (&MYBI.MoveNext());

Else

  /* Process error - no output buffer */

End-If;
```

Related Topics

AddInputRow, Execute, MoveNext, GetFieldCount, GetFieldType, GetFieldValue, Supporting Standard Input/Output: BIDocs Methods

MoveNext

Syntax

```
MoveNext()
```

Description

The **MoveNext** method is used to support dynamic output. This method places you in the next row of the output buffer. You can only use this method after you have used the **MoveFirst** method: otherwise the system doesn't know where to start.

Parameters

None.

Returns

Boolean: True if successfully positioned cursor at the next row of the output buffer, False otherwise.

Example

In the following example, the Business Interlink Object name is &MYBI. The example uses **MoveFirst** to move to the first row of the output buffer, and to the first column, or first field, in that row. The **Repeat** loop uses **MoveNext** to go through every row in the output buffer. The **For** loop processes every field in every row, using the **GetFieldCount** method to get the number of fields, or outputs, in the row.

```
/* Add inputs to the Business Interlink Object, then call Execute to execute the
Business Interlink Object. You are then ready to get the outputs using the
following code. */
```

```
If (&MYBI.MoveFirst()) Then

  Repeat

    For &I = 1 to &MYBI.GetFieldCount

      &TYPE = &MYBI.GetFieldType(&I);

      Evaluate &TYPE

      Where = 1

      &STRING_VARIABLE = &MYBI.GetFieldValue(&I);
```

```
        /* test for and process other field types */  
  
        End-Evaluate;  
  
    End-For;  
  
    Until Not (&MYBI.MoveNext());  
  
    Else  
  
        /* Process error - no output buffer */  
  
    End-If;
```

Related Topics

AddInputRow, Execute, MoveFirst, GetFieldCount, GetFieldType, GetFieldValue, Supporting Standard Input/Output: BIDocs Methods

BIDocs Hierarchical Methods

This section describes the BIDocs methods that are used for hierarchical input and output data.



For more information about the BIDocs in general, see Supporting Standard Input/Output: BIDocs Methods.



The Business Interlink methods GetFieldCount, GetFieldType, and GetFieldValue can also be used with BIDocs input and output data.

AddDoc

Syntax

AddDoc (*docname*)

Description

The **AddDoc** method adds a document to a BIDocs input document. The added document is an input parameter for a Business Interlink Object that is not of simple type (such as integer or string). You must add the document to the BIDocs input document before you can add values to its members with **AddValue**.

Parameters

Docname The name of the document that **AddDoc** adds to the BIDocs document.

Return Value

The document added to the BIDocs input document.

Example

In the following example, the BIDocs input document for Calculate Cost, or the root level document, is created with the **GetInputDocs** method. (If you wanted to create, or add, more BIDocs input documents, you would call **AddNextDoc**.) The Calculate Cost input parameter From is a document, so the **AddDoc** method adds it to the input document.

```
Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostIn;

Local BIDocs &FromDoc, &ToDoc, &PackageDoc, &AccountDoc;

Local number &ret, &retinput;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

&CalcCostIn = &QE_FEDEX_COST.GetInputDocs("");

&ret = &CalcCostIn.AddValue("input_param1", "value");

&FromDoc = &CalcCostIn.AddDoc("From");

&ret = &FromDoc.AddValue("OriginCountry", "United States");

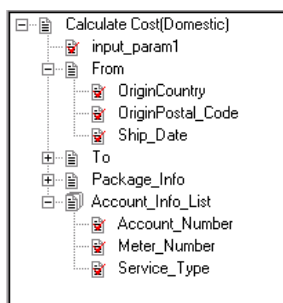
&ret = &FromDoc.AddValue("OriginPostal_Code", &ORIGIN);

&ret = &FromDoc.AddValue("Ship_Date", &SHIPDATE);

/* Call AddDoc and AddValue for To, Package_Info, and Account_Info_List (code
not shown) */
```

The figure below shows the BIDocs input document for this example. It contains five input parameters: input_param1, From, To, Package_Info, and Account_Info_List. This is a version of

the Federal Express plug-in that was modified for this example (input_param1 was added, Account_Info was modified to be a list).



Example BIDocs Input Document

Related Topics

AddNextDoc, GetInputDocs

AddNextDoc

Syntax

```
AddNextDoc ( )
```

Description

The **AddNextDoc** method adds a document to one of the following levels:

- The root level of the BIDocs input document for a Business Interlink Object. This level was created with the method **GetInputDocs**.
- When a document within the BIDocs input document is a list, **AddNextDoc** adds another document to the list. If you use **AddNextDoc** on a document that is not a list, **AddNextDoc** fails and returns an error.

Parameters

None.

Return Values

<i>Number</i>	<i>Enum value and Meaning</i>
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.

2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

Example

In this example, the BIDocs input document for Calculate Cost, or the root level document, is created with the **GetInputDocs** method. The **AddNextDoc** method adds another BIDocs input document. The Calculate Cost input parameter Account_Info_List is a document, so the **AddDoc** method adds it to the BIDocs input document. Account_Info_List is also a document list, so **AddNextDoc** method adds another Account_Info_List document.

```

Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostIn;

Local BIDocs &FromDoc, &ToDoc, &PackageDoc, &AccountDoc;

Local number &ret, &retinput;


&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);


&CalcCostIn = &QE_FEDEX_COST.GetInputDocs("");


For &n = 1 to &number_of_input_sets

/* Get values for inputs, such as &ORIGIN (code not shown) */

```

```

&ret = &CalcCostIn.AddValue("input_param1", "value");

&FromDoc = &CalcCostIn.AddDoc("From");

&ret = &FromDoc.AddValue("OriginCountry", "United States");

&ret = &FromDoc.AddValue("OriginPostal_Code", &ORIGIN);

&ret = &FromDoc.AddValue("Ship_Date", &SHIPDATE);


/* Call AddDoc and AddValue for To and Package_Info
   (code not shown) */


/* Call AddDoc and AddNextDoc for the AccountDoc document list */
&AccountDoc = &CalcCostIn.AddDoc("Account_Info_List");
For &m = 1 to &number_of_AccountDocs

    &ret = &AccountDoc.AddValue("Account_Number", &ACCOUNT);

    &ret = &AccountDoc.AddValue("Meter_Number", &METER);

    &ret = &AccountDoc.AddValue("Service_Type", &SVCTYPE);

    &retinput = &AccountDoc.AddNextDoc();

End-For;


&retinput = &CalcCostIn.AddNextDoc();

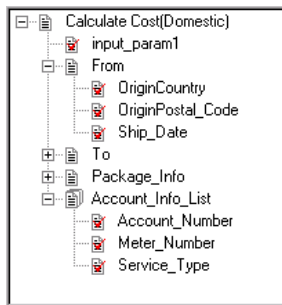

End-For;

```

The figure below shows the BIDocs input document for this example. It contains five input parameters: input_param1, From, To, Package_Info, and Account_Info_List. This is a version of the Federal Express plug-in that was modified for this example (input_param1 was added, Account_Info was modified to be a list).

From, To, and Package_Info are not lists, so if you try to use **AddNextDoc** with these parameters, such as the following line of code, **AddNextDoc** will fail and return an error message.

```
&retinput = &To.AddNextDoc();
```



Example BIDocs Input Document

Related Topics

AddDoc, GetInputDocs

AddValue

Syntax

```
AddValue(paramname, value)
```

Description

The **AddValue** method adds a value to an input parameter within a BIDocs document for a Business Interlink Object.

Parameters

<i>paramname</i>	The name of the member of the document that is having a value added to it.
<i>value</i>	The value that is added. This can be a constant, a variable, or a record field. The data type can be string, integer, double, float, time, date, or datetime.

Return Values

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.
2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.

4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

Example

In the following example, the Business Interlink Object name is QE_FEDEX_COST.

In the following example, the BIDocs input document for Calculate Cost, or the root level document, is created with the **GetInputDocs** method. (If you wanted to create, or add, more BIDocs input documents, you would call **AddNextDoc**.) The Calculate Cost input parameter From is a document, so the **AddDoc** method adds it to the BIDocs input document. Then the **AddValue** method adds values to each of the From document members.

```

Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostIn;

Local BIDocs &FromDoc, &ToDoc, &PackageDoc, &AccountDoc;

Local number &ret;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

/* Get some values for input, such as &ORIGIN (code not shown) */

&CalcCostIn = &QE_FEDEX_COST.GetInputDocs("");

&ret = &CalcCostIn.AddValue("input_param1", "value");

&FromDoc = &CalcCostIn.AddDoc("From");

&ret = &FromDoc.AddValue("OriginCountry", "United States");

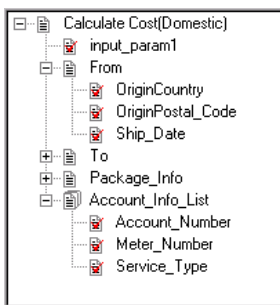
&ret = &FromDoc.AddValue("OriginPostal_Code", &ORIGIN);

```

```
&ret = &FromDoc.AddValue("Ship_Date", &SHIPDATE);
```

```
/* Call AddDoc, AddValue for Package, Account_Info_List, and also call  
AddNextDoc for Account_Info_List (code not shown) */
```

The figure below shows the BIDocs input document for this example. It contains five input parameters: input_param1, From, To, Package_Info, and Account_Info_List. This is a version of the Federal Express plug-in that was modified for this example (input_param1 was added, Account_Info was modified to be a list).



Example BIDocs Input Document

Related Topics

AddDoc, AddNextDoc, GetInputDocs

Clear

Syntax

```
Clear()
```

Description

The **Clear** method clears the BIDocs document for a Business Interlink Object. If you're using Business Interlinks in a batch program, every time after you use the **Execute** method, you want to use the **Clear** method to flush out the BIDocs document.

Parameters

None.

Returns

None.

Example

In the following example, the BIDocs output document for Calculate Cost, or the root level document, is created with the **GetOutputDocs** method. The Calculate Cost output parameter output_param2_List is a document, so the **GetDoc** method gets it from the BIDocs output document. Since output_param2_List is a document list, **GetCount** gets the number of documents in the list.

After the output_param2_List document is processed, the BIDocs document is cleared with **Clear**.

```

Local Interlink &QE_FEDEX_COST;

Local number &count;

Local BIDocs &CalcCostOut;

Local BIDocs &OutlistDoc;

Local number &ret;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");
&OutlistDoc = &CalcCostOut.GetDoc("output_param2_List");

&count = &CalcCostOut.GetCount("output_param2_List");

&I = 0;

While (&I < &count)

    &ret = &OutlistDoc.GetValue("output_member1", &VALUE1);

    &ret = &OutlistDoc.GetValue("output_member2", &VALUE2);

    If &ret = 0 Then

        /* Process output values */

        &I = &I + 1;

        &retoutput = &OutlistDoc.GetNextDoc();

    End-If;

End-While;

```

```
&CalcCostOut.Clear();
```

GetCount

Syntax

GetCount (*docname*)

Description

The **GetCount** method returns the number of documents within a document list or parameter list contained within a BIDocs output document for a Business Interlink Object.

Parameters

<i>docname</i>	The name of the document list or parameter list.
----------------	--

Return Value

The number of documents in the list.

Example

In the following example, the BIDocs output document for Calculate Cost, or the root level document, is created with the **GetOutputDocs** method. The Calculate Cost output parameter `output_param2_List` is a document, so the **GetDoc** method gets it from the BIDocs output document. Since `output_param2_List` is a document list, **GetCount** gets the number of documents in the list.

```
Local Interlink &QE_FEDEX_COST;

Local number &count;

Local BIDocs &CalcCostOut;

Local BIDocs &OutlistDoc;

Local number &ret;


&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)


&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");
```

```

/* Call GetValue for output_param1, call GetDoc, GetValue for Service_Rate (code
not shown) */

&OutlistDoc = &CalcCostOut.GetDoc("output_param2_List");

&count = &CalcCostOut.GetCount("output_param2_List");

&I = 0;

While (&I < &count)

    &ret = &OutlistDoc.GetValue("output_member1", &VALUE1);

    &ret = &OutlistDoc.GetValue("output_member2", &VALUE2);

    If &ret = 0 Then

        /* Process output values */

        &I = &I + 1;

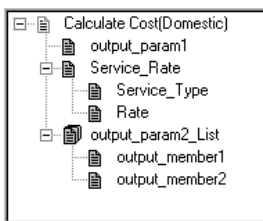
        &retoutput = &OutlistDoc.GetNextDoc();

    End-If;

End-While;

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: output_param1, Service_Rate, and output_param2_List. This is a version of the Federal Express plug-in that was modified for this example (output_param1 and output_param2_List were added).



Example BIDocs Output Document

GetDoc

Syntax

GetDoc (*docname*)

Description

The **GetDoc** method gets a document from a BIDocs output document. The document is an output parameter for a Business Interlink Object that is not of simple type (such as integer or string). You must get the document from the BIDocs output document before you can get values from its members with **GetValue**.

You can call **GetDoc** using a nesting feature. This feature allows you to access deeply nested documents with one call to **GetDoc**, rather than calling **GetDoc** for each nesting. See the Example section below for an example of this.

Parameters

<i>docname</i>	The name of the document that GetDoc gets from the BIDocs document.
----------------	--

Return Value

The document received from the BIDocs output document.

Example

In the following example, the BIDocs output document for Calculate Cost(Domestic), or the root level document, is created with the **GetOutputDocs** method. (If you wanted to create, or get, more BIDocs output documents, you would call **GetNextDoc**.) The Calculate Cost output parameter Service_Rate is a document, so the **GetDoc** method gets it from the BIDocs output document.

```
Local Interlink &QE_FEDEX_COST;

Local number &count;

Local BIDocs &CalcCostOut;

Local BIDocs &ServiceRateDoc;

Local number &ret;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

&ret = &CalcCostOut.GetValue("output_param1",&VALUE);

&ServiceRateDoc = &CalcCostOut.GetDoc("Service_Rate");

&ret = &ServiceRateDoc.GetValue("Service_Type", &SERVICE_TYPE);

&ret = &ServiceRateDoc.GetValue("Rate", &RATE);
```

```

/* Call GetDoc, GetValue, GetNextDoc for output_param2_List (code not shown) */

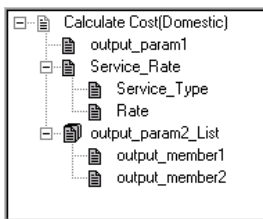
If &ret = 0 Then

    /* Process output values */

End-If;

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: output_param1, Service_Rate, and output_param2_List. This is a version of the Federal Express plug-in that was modified for this example (output_param1 and output_param2_List were added).



Example BIDocs Output Document

In the following example, **GetDoc** is used to access a document that is nested more deeply. If you want to access a document that is more deeply nested, you can either call **GetDoc** for each nesting, or you can call **GetDoc** once using the nesting feature.

Calling **GetDoc** with the nesting feature:

```

Local Interlink &QE_4GETDOC;

Local BIDocs &CalcCostOut, &Docs3;

Local number &ret;

&QE_4GETDOC = GetInterlink(Interlink.QE_4GETDOC_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_4GETDOC.GetOutputDocs("");

&Docs3 = &CalcCostOut.GetDoc("Doc1.Doc2.Doc3");

&ret = &Docs3.GetValue("output_member3", &VALUE);

```

Calling **GetDoc** without the nesting feature:

```

Local Interlink &QE_4GETDOC;

```

```

Local BIDocs &CalcCostOut, &Docs1, &Docs2, &Docs3;

Local number &ret;

&QE_4GETDOC = GetInterlink(Interlink.QE_4GETDOC_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_4GETDOC.GetOutputDocs("");

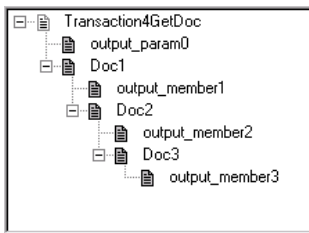
&Docs1 = &CalcCostOut.GetDoc("Doc1");

&Docs2 = &Docs1.GetDoc("Doc2");

&Docs3 = &Docs2.GetDoc("Doc3");

&ret = &Docs3.GetValue("output_member3", &VALUE);

```



Example BIDocs Output Document: Nested Documents

GetNextDoc

Syntax

```
GetNextDoc ( )
```

Description

The **GetNextDoc** method gets a document from one of the following levels:

- The root level of the BIDocs output document for a Business Interlink Object. This level was created with the method **GetOutputDocs**.
- When a document within the BIDocs output document is a list, **GetNextDoc** gets another document from the list. If you use **GetNextDoc** on a document that is not a list, **GetNextDoc** fails and returns an error.

Parameters

None.

Return Values

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.
2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

Example

In this example, the BIDocs output document for Calculate Cost, or the root level document, is created with the **GetOutputDocs** method. The **GetNextDoc** method gets another BIDocs output document, assuming that there is more than one of them. The Calculate Cost output parameter output_param2_List is a document, so the **GetDoc** method gets it to the BIDocs output document. output_param2_List is also a document list, so **GetNextDoc** method gets the next output_param2_List document.

```
Local Interlink &QE_FEDEX_COST;

Local number &count1, &count2;

Local BIDocs &CalcCostOut;

Local BIDocs &OutlistDoc;

Local number &ret1, &ret2;
```

```

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

&ret1 = 0;

While (&ret1)

    &ret1 = &CalcCostOut.GetValue("output_param1",&VALUE);

    &ServiceRateDoc = &CalcCostOut.GetDoc("Service_Rate");

    &ret1 = &ServiceRateDoc.GetValue("Service_Type", &SERVICE_TYPE);

    &ret1 = &ServiceRateDoc.GetValue("Rate", &RATE);

    /* Process output values */

    &OutlistDoc = &CalcCostOut.GetDoc("output_param2_List");

    &count2 = &CalcCostOut.GetCount("output_param2_List");

    While (&I < &count2)

        &ret2 = &OutlistDoc.GetValue("output_member1", &VALUE1);

        &ret2 = &OutlistDoc.GetValue("output_member2", &VALUE2);

        If &ret2 = 0 Then

            /* Process output values */

            &I = &I + 1;

            &ret2 = &OutlistDoc.GetNextDoc();

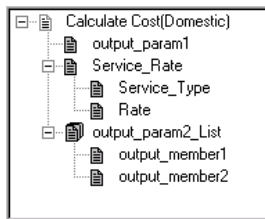
        End-If;

        &ret1 = &CalcCostOut.GetNextDoc();

    End-While;

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: output_param1, Service_Rate, and output_param2_List. This is a version of the Federal Express plug-in that was modified for this example (output_param1 and output_param2_List were added).



Example BIDocs Output Document

GetPreviousDoc

Syntax

```
GetPreviousDoc()
```

Description

The **GetPreviousDoc** method gets the previous document from either the root level of the BIDocs output document for a Business Interlink Object, or from a document within the BIDocs output document. When these documents are in a list, **GetPreviousDoc** gets the previous document in the list. You must get the document before you can get its values with **GetValue**.

Parameters

None.

Return Values

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.
2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.

- | | |
|----|---|
| 9 | NO_DATA. You tried to retrieve data from a document that contained no data. |
| 10 | GENERIC_ERROR. There was an error with the transaction. |

Example

In this example, the BIDocs output document for Calculate Cost, or the root level document, is created with the **GetOutputDocs** method. It contains one output parameter, output_param2_List, which is also a list. The **GetCount** and **MoveToDoc** methods point to the last output_param2_List document in the list. The **GetPreviousDoc** method is used in a loop to cycle through the output_param2_List list, starting with the last and ending with the first in the list, and get each output_param2_List document and its values.

```

Local Interlink &QE_FEDEX_COST;

Local number &count;

Local BIDocs &CalcCostOut;

Local BIDocs &OutlistDoc;

Local number &ret;

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

/* Call GetValue for output_param1, call GetDoc, GetValue for Service_Rate (code
not shown) */

&OutlistDoc = &CalcCostOut.GetDoc("output_param2_List");

&count = &CalcCostOut.GetCount("output_param2_List");

&ret = &OutlistDoc.MoveToDoc(&count-1);

&I = &count;

While (&I > 0)

    &ret = &OutlistDoc.GetValue("output_member1", &VALUE1);

    &ret = &OutlistDoc.GetValue("output_member2", &VALUE2);

    If &ret = 0 Then

```

```

/* Process output values */

&I = &I - 1;

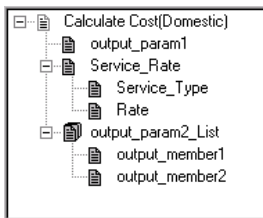
&retoutput = &OutlistDoc.GetPreviousDoc("");

End-If;

End-While;

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: output_param1, Service_Rate, and output_param2_List. This is a version of the Federal Express plug-in that was modified for this example (output_param1 and output_param2_List were added).



Example BIDocs Output Document

GetStatus

Syntax

```
GetStatus()
```

Description

The **GetStatus** method tests the BIDocs document. To find the status for any BIDocs method that does not return a status value, call GetStatus just after you call that BIDocs method.

Parameters

None.

Return Value

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.

2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

GetValue

Syntax

```
GetValue(paramname, value)
```

```
Number GetValue(paramname, value)
```

Description

The **GetValue** method gets a value from an output parameter within a BIDocs output document for a Business Interlink Object.

Parameters

<i>paramname</i>	The name of the member of the document that is having a value retrieved from it.
<i>value</i>	The value that is retrieved. This can be a variable or a record field. The data type can be string, integer, double, float, time, date, or datetime.

Return Value

Value	Meaning
string, <i>value</i>	The value of the output parameter.

Number	When the syntax is <code>Number GetValue(name, value)</code> , number is the return status of <code>GetValue</code> , listed in the table below.
--------	--

Return Value for integer

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.
2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call <code>GetDoc/GetOutputDocs</code> once, you can call <code>GetNextDoc</code> four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a <code>GetNextDoc</code> or <code>AddNextDoc</code> upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using <code>GetValue</code> , <code>AddValue</code> , on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a <code>GetValue</code> or <code>AddValue</code> upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

Example

In the following example, the Business Interlink Object name is `QE_FEDEX_COST`.

In the following example, the `BIDocs` output document for Calculate Cost, or the root level document, is created with the **`GetOutputDocs`** method. (If you wanted to create, or get, more `BIDocs` output documents, you would call **`GetNextDoc`**.) The Calculate Cost output parameter `Service_Rate` is a document, so the **`GetDoc`** method gets it from the `BIDocs` output document. Then the **`GetValue`** method gets values from each of the `Service_Rate` document members.

```
Local Interlink &QE_FEDEX_COST;

Local BIDocs &CalcCostOut;

Local BIDocs &ServiceRateDoc;

Local number &ret;
```

```

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

&ret = &CalcCostOut.GetValue("output_param1",&PARAM1);

&ServiceRateDoc = &CalcCostOut.GetDoc("Service_Rate");

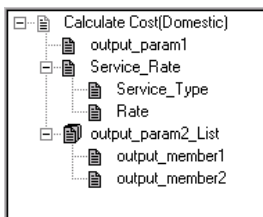
&ret = &ServiceRateDoc.GetValue("Service_Type", &SERVICE_TYPE);

&ret = &ServiceRateDoc.GetValue("Rate", &RATE);

/* Call GetDoc, GetValue, GetNextDoc for output_param2_List (code not shown) */

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: `output_param1`, `Service_Rate`, and `output_param2_List`. This is a version of the Federal Express plug-in that was modified for this example (`output_param1` and `output_param2_List` were added).



Example BIDocs Output Document

MoveToDoc

Syntax

```
MoveToDoc (list_number)
```

Description

Within a list of documents in the BIDocs output document, the **MoveToDoc** method moves to the documents given by the parameter *list_number*. After using **MoveToDoc**, the **GetValue** method gets the values of the document that is in the *list_number*+1 location in the list.

Parameters

list_number

The number indicating the document that **MoveToDoc** moves to. After using **MoveToDoc**, the **GetValue** method gets the values of the document that is in the *list_number*+1 location in the list. For example, if *list_number* is zero, then **MoveToDoc** moves to the first document in the list.

Return Values

Number	Enum value and Meaning
0	NoError. The method succeeded.
1	NO_DOCUMENT. The document referenced by this method does not exist.
2	LIST_OUT_RANGE. You tried to access a document in a list, but the document list is out of range. For example, if a document list contains five documents, and then you call GetDoc/GetOutputDocs once, you can call GetNextDoc four times; the fifth time will result in this error.
3	DOCUMENT_UNINITIALIZED. Internal error.
4	NOT_DOCUMENTTYPE. You tried to perform an operation upon a parameter that is not a document type.
5	NOT_DOCUMENTLISTTYPE. You tried to perform a GetNextDoc or AddNextDoc upon a document that is not a list.
6	NOT_LISTTYPE. You tried to perform a list operation using GetValue, AddValue, on a non-list.
7	NOT_SINGLEBASICTYPE. You tried to perform a GetValue or AddValue upon a list that does not use a single basic type: integer, float, string, time, date, datetime.
9	NO_DATA. You tried to retrieve data from a document that contained no data.
10	GENERIC_ERROR. There was an error with the transaction.

Example

The following example gets the values of the last document in the output_param2_List list. It uses **GetCount** to get the number of documents in the list, and then uses **MoveToDoc** to move to the last document in the list.

```
Local Interlink &QE_FEDEX_COST;

Local number &count;

Local BIDocs &CalcCostOut;

Local BIDocs &OutlistDoc;

Local number &ret;
```

```

&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);

// Get inputs, execute. (code not shown)

&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");

&OutlistDoc = &CalcCostOut.GetDoc("output_param2_List");

&count = &CalcCostOut.GetCount("output_param2_List");

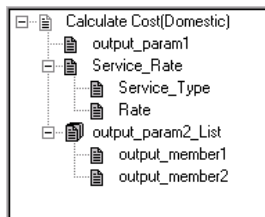
&ret = &OutlistDoc.MoveToDoc(&count-1);

&ret = &OutlistDoc.GetValue("output_member1", &VALUE1);

&ret = &OutlistDoc.GetValue("output_member2", &VALUE2);

```

The figure below shows the BIDocs output document for this example. It contains three output parameters: output_param1, Service_Rate, and output_param2_List. This is a version of the Federal Express plug-in that was modified for this example (output_param1 and output_param2_List were added).



Example BIDocs Output Document

ResetCursor

Syntax

```
ResetCursor()
```

Description

The **ResetCursor** method resets the cursor in the BIDocs output document for a Business Interlink Object to the top. Once you call this method, the next time you call **GetValue**, you get an output value from the first document of the BIDocs output documents for a Business Interlink Object.

Parameters

None.

Return Value

None.

Example

The following code uses **ResetCursor** to reset the cursor in the BIDocs output document to the top.

```
Local Interlink &QE_FEDEX_COST;  
  
Local BIDocs &CalcCostOut;  
  
&QE_FEDEX_COST = GetInterlink(Interlink.QE_FEDEX_COST_EX);  
  
// Get inputs, execute. (code not shown)  
  
&CalcCostOut = &QE_FEDEX_COST.GetOutputDocs("");  
  
// Perform actions on the output documents (code not shown)  
  
&CalcCostOut.ResetCursor();
```

Incoming Business Interlink BIDocs Methods And Properties

This section describes the PeopleCode methods and functions you use with the Incoming Business Interlink; they parse an XML request contained within a BiDocs object and build an XML response within a BiDocs object. These are all BiDocs methods.

AddAttribute

Syntax

```
AddAttribute(attributename, attributevalue)
```

Description

The **AddAttribute** method adds an attribute name and its value to an XML element referenced by a BiDocs object.

Parameters

<i>attributename</i>	String. The name of the attribute.
<i>attributevalue</i>	String. The value of the attribute.

Return Value

Number. The return status. NoError, or 0, means the method succeeded.



For more information on the return status, see `GetStatus`.

Example

Here is a set of XML response code.

```
<?xml version="1.0"?>

<postreqresponse>

  <candidate>

    <user>

      <location scenery="great" density="low" blank="eh?">

    </location>

    </user>

  </candidate>

</postreqresponse>
```

Here is the PeopleCode that builds it.

```
Local BIDocs &rootDoc, &postreqresponse;

Local BIDocs &candidates, &location, &user;

Local number &ret;

&rootDoc = GetBiDoc("");

&ret = &rootDoc.AddProcessInstruction("<?xml version=\"1.0\"?>");
```

```
&postreqresponse = &rootDoc.CreateElement("postreqresponse");  
  
&candidates = &postreqresponse.CreateElement("candidates");  
  
&user = &candidates.CreateElement("user");  
  
&location = &user.CreateElement("location");  
  
&ret = &location.AddAttribute("scenery", "great");  
  
&ret = &location.AddAttribute("density", "low");  
  
&ret = &location.AddAttribute("blank", "eh?");
```

AddComment

Syntax

AddComment (*comment*)

Description

The **AddComment** method adds an XML comment after the beginning tag of an XML element referenced by a BiDoc object.

Parameters

<i>comment</i>	String. The comment.
----------------	----------------------

Return Value

Number. The return status. NoError, or 0, means the method succeeded.



For more information on the return status, see GetStatus.

Example

Here is a set of XML response code.

```
<?xml version="1.0"?>  
  
<postreqresponse>  
  
<error>  
  
<!--this is a comment line-->  
  
<errorcode>1</errorcode>  
  
<errortext></errortext>
```



```

        </error>

    </postreqresponse>

```

Here is the PeopleCode that builds it.

```

Local BIDocs &rootDoc, &postreqresponse, &error, &errorcode, &errortext;

Local string &blob;

Local number &ret;

&rootDoc = GetBiDoc("");

/* add a processing instruction*/

&ret = &rootDoc.AddProcessInstruction("<?xml version='1.0'?>");

/* create an element and add text*/

&postreqresponse = &rootDoc.CreateElement("postreqresponse");

&error = &postreqresponse.CreateElement("error");

&ret = &error.AddComment("this is a comment line");

&errorcode = &error.CreateElement("errorcode");

&ret = &errorcode.AddText("1");

&errortext = &error.CreateElement("errortext");

```

AddProcessInstruction

Syntax

```
AddProcessInstruction(instruction)
```

Description

The **AddProcessInstruction** method adds an XML processing instruction to a BiDocs object. Use this method at the root level of the BiDocs object for Incoming Business Interlinks before you add anything else to the BiDocs object.

Parameters

<i>instruction</i>	String. The processing instruction.
--------------------	-------------------------------------

Return Value

Number. The return status. NoError, or 0, means the method succeeded.



For more information on the return status, see `GetStatus`.

Example

Here is a set of XML response code.

```
<?xml version="1.0"?>

<postreqresponse>

    <error>

        <!--this is a comment line-->

        <errorcode>1</errorcode>

        <errortext></errortext>

    </error>

</postreqresponse>
```

Here is the PeopleCode that builds it.

```
Local BIDocs &rootDoc, &postreqresponse;

Local BIDocs &error, &errorcode, &errortext;

Local number &ret;

&rootDoc = GetBiDoc("");

/* add a processing instruction*/

&ret = &rootDoc.AddProcessInstruction("<?xml version=\"1.0\"?>");

/* create an element and add text*/

&postreqresponse = &rootDoc.CreateElement("postreqresponse");

&error = &postreqresponse.CreateElement("error");

&ret = &error.AddComment("this is a comment line");

&errorcode = &error.CreateElement("errorcode");

&ret = &errorcode.AddText("1");

&errortext = &error.CreateElement("errortext");
```

AddText

Syntax

AddText (*text*)

Description

The **AddText** method adds text to an XML element referenced by a BiDocs object.

Parameters

<i>text</i>	String. The text.
-------------	-------------------

Return Value

Number. The return status. NoError, or 0, means the method succeeded.



For more information on the return status, see [GetStatus](#).

Example

Here is a set of XML response code.

```
<?xml version="1.0"?>

<postreqresponse>

    <error>

        <!--this is a comment line-->

        <errorcode>1</errorcode>

        <errortext></errortext>

    </error>

</postreqresponse>
```

Here is the `PeopeCode` that builds it.

```
Local BIDocs &rootDoc, &postregresponse;  
  
Local BIDocs &error, &errorcode, &errortext;  
  
Local number &ret;  
  
&rootDoc = GetBiDoc("");
```

```

/* add a processing instruction*/

&ret = &rootDoc.AddProcessInstruction("<?xml version=\"1.0\"?>");

/* create an element and add text*/

&postreqresponse = &rootDoc.CreateElement("postreqresponse");

&error = &postreqresponse.CreateElement("error");

&ret = &error.AddComment("this is a comment line");

&errorcode = &error.CreateElement("errorcode");

&ret = &errorcode.AddText("1");

&errortext = &error.CreateElement("errortext");

```

AttributeCount

Syntax

```
AttributeCount
```

Description

The **AttributeCount** property gets the number of attributes within an XML element referenced by a BiDocs object. This property is read-only.

Parameters

None.

Return Value

Number. The number of attributes.

Example

Here is a set of XML request code.

```

<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <location scenery="great" density="low" blank="eh?">

    <city>San Rafael</city>

    <state>CA</state>

    <zip>94522</zip>

```

```

        <country>US</country>

    </location>

</postreq>

```

Here is the PeopleCode that gets the number of attributes in the location XML element. &count should be 3, for scenery, density, and blank.

```

Local BiDocs &rootInDoc, &postreqDoc, &locationDoc;

Local string &blob;

Local number &count;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode("postreq");

&locationDoc = &postreqDoc.GetNode("location");

&count = &locationDoc.AttributeCount;

```

ChildNodeCount

Syntax

```
ChildNodeCount
```

Description

The **ChildNodeCount** property returns the number of XML child nodes within the element referenced by the BiDocs object. Child nodes include XML elements, comments, and processing instructions. This property is read-only.

Parameters

None.

Return Value

Number. The number of child nodes.

Example

Here is a set of XML request code.

```

<?xml version="1.0"?>

    <postreq>

```

```

    <email>joe_blow@peoplesoft.com</email>

    <projtitle>

        <!--this is a comment line-->

        <projsubtitle>first_subtitle</projsubtitle>

        <projsubtitle>second_subtitle</projsubtitle>

        <projsubtitle>third_subtitle</projsubtitle>

    </projtitle>

</postreq>

```

Here is the XML code that gets the number of nodes within <postreq> and <projtitle>. &count1 is 2, for <email> and <projtitle>, and &count2 is 4, for the three <projsubtitle> nodes and the comment node.

```

Local BIDocs &rootInDoc, &projtitleDoc;

Local string &blob;

Local number &count1, &count2;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode("postreq");

&count1 = &postreqDoc.ChildNodeCount;

&projtitleDoc = &postreqDoc.GetNode("projtitle");

&count2 = &projtitleDoc.ChildNodeCount;

```

CreateElement

Syntax

```
CreateElement(elementname)
```

Description

The **CreateElement** method creates an XML element with the given name within a BiDoc object.

Parameters

elementname String. The XML element name.

Return Value

BiDocs. The reference to the created element.

Example

Here is a set of XML response code.

```
<?xml version="1.0"?>

  <postreqresponse>

    <error>

      <errorcode>1</errorcode>

      <errortext></errortext>

    </error>

  </postreqresponse>
```

Here is the PeopleCode that builds it.

```
Local BiDocs &rootDoc, &postreqresponse;

Local BiDocs &error, &errorcode, &errortext;

Local number &ret;

&rootDoc = GetBiDoc("");

/* add a processing instruction*/
&ret = &rootDoc.AddProcessInstruction("<?xml version=\"1.0\"?>");

/* create an element and add text*/

&postreqresponse = &rootDoc.CreateElement("postreqresponse");

&error = &postreqresponse.CreateElement("error");

&errorcode = &error.CreateElement("errorcode");

&ret = &errorcode.AddText("1");

&errortext = &error.CreateElement("errortext");
```

GenXMLString

Syntax

```
GenXMLString()
```

Description

The **GenXMLString** method creates an XML string from a BiDocs object. The BiDocs object must contain the shape and data needed for an XML string. This is part of the Incoming Business Interlink functionality, which allow PeopleCode to receive an XML request and return an XML response.

Parameters

None.

Return Value

String. The XML string containing the shape and data of the BiDocs object. For example, you can use this method to create an XML string containing an XML response.

Example

The following example takes a BiDocs structure that contains an XML response and puts that into a text string. Once this is done, the %Response.Write function can send this as an XML response.

```
Local BiDocs &rootDoc;

Local string &xmlString;

/* Create a BiDoc structure containing the data and shape of your XML response
(code not shown) */

/* Generate the XML string containing the data and shape of your XML response
from the BiDoc structure */

&xmlString = &rootDoc.GenXMLString();

%Response.Write(&xmlString);
```

GetAttributeName

Syntax

```
GetAttributeName(attributenum)
```


Description

The **GetAttributeName** method gets the name of an attribute within an XML element referenced by a BiDocs object.

Parameters

attributenum Number. The index number of the attribute.

Return Value

String. The name of the attribute.

Example

Here is a set of XML request code.

```
<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <location scenery="great" density="low" blank="eh?">

    <city>San Rafael</city>

    <state>CA</state>

    <zip>94522</zip>

    <country>US</country>

  </location>

</postreq>
```

Here is the PeopleCode that gets the name of the second attribute in the location node. &attrName is density.

```
Local BiDocs &rootInDoc, postreqDoc, &locationDoc;

Local string &blob, &attrName;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode("postreq");

&locationDoc = &postreqDoc.GetNode("location");

&attrName = &locationDoc.GetAttributeName(2);
```

GetAttributeValue

Syntax

```
GetAttributeValue({attributename | attributenum})
```

Description

The **GetAttributeValue** method gets the value of an attribute within an XML element referenced by a BiDocs object.

Parameters

<i>attributenum</i> <i>attributename</i>	Specify the attribute you want to get the value for. You can specify either the attribute number (1 for the first attribute, 2 for the second, and so on) or the attribute name (an XML tag.)
---	---

Return Value

String. The value of the attribute.

Example

Here is a set of XML request code.

```
<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <location scenery="great" density="low" blank="eh?">

    <city>San Rafael</city>

    <state>CA</state>

    <zip>94522</zip>

    <country>US</country>

  </location>

</postreq>
```

Here is the PeopleCode that gets the value of the second attribute in the location node. &attrValue is low.

```
Local BiDocs &rootInDoc, &postreqDoc, &locationDoc;

Local string &blob, &attrValue;

&blob = %Request.GetContentBody();
```

```

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode("postreq");

&locationDoc = &postreqDoc.GetNode("location");

&attrValue = &locationDoc.GetAttributeValue(2);

```

GetNode

Syntax

```
GetNode({nodename | nodenumber})
```

Description

The **GetNode** method returns a BiDocs reference to a child XML node (element or comment). Use the **GetNode** method upon a BiDocs reference to an XML element in order to access the child nodes for that element.

Parameters

<i>nodenumber</i> <i>nodename</i>	Specify the node you want to reference. You can specify either a node number (the first node is 1, the second 2, and so on) or a node name (that is, the XML tag.)
-------------------------------------	--

Return Value

BiDocs. The returned XML element within a BiDocs object.

Example

Here is a set of XML request code.

```

<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <projtitle>

    <projsubtitle>first_subtitle</projsubtitle>

    <projsubtitle>second_subtitle</projsubtitle>

    <projsubtitle>third_subtitle</projsubtitle>

  </projtitle>

</postreq>

```

Here is the PeopleCode that gets the postreqDoc element and the projtitle element.

```
Local BIDocs &rootInDoc, &postreqDoc, &projtitleDoc;

Local string &name, &blob;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode("postreq");

&projtitleDoc = &postreqDoc.GetNode("projtitle");
```

Here is the PeopleCode that gets the postreqDoc element, the projtitle element, and the third projsubtitle element.

```
Local BIDocs &rootInDoc, &postreqDoc, &projtitleDoc, &projsubtitleDoc;

Local string &name, &blob;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode(1);

&projtitleDoc = &postreqDoc.GetNode(2);

&projsubtitleDoc = &projtitleDoc.GetNode(3);
```

In order to parse a list of elements like <projsubtitle>, where elements have the same name, you must call **GetNode** using an index number rather than the element name.

NodeName

Syntax

```
NodeName
```

Description

The **NodeName** property gets the name of an XML element referenced by a BiDocs object. Use this to get the name of an XML element when you used **GetNode** with an index number to retrieve it (meaning that you did not have the name of the XML element when you used **GetNode**). This property is read-only.

Parameters

None.

Return Value

String. The node name.

Example

Here is a set of XML request code.

```
<?xml version="1.0"?>

  <postreq>

    <email>joe_blow@peoplesoft.com</email>

    <projtitle>

      <projsubtitle>first_subtitle</projsubtitle>

      <projsubtitle>second_subtitle</projsubtitle>

      <projsubtitle>third_subtitle</projsubtitle>

    </projtitle>

  </postreq>
```

Here is the PeopleCode that gets the name of the <email> element, email.

```
Local BiDocs &rootInDoc, &postreqDoc, &emailDoc;

Local string &emailName, &blob;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode(1);

&emailDoc = &postreqDoc.GetNode(1);

&emailName = &emailDoc.NodeName;
```

NodeType

Syntax

NodeType

Description

The **NodeType** property gets the type of an XML element referenced by a BiDocs object. This property is read-only.

Parameters

None.

Return Value

Number. The node type.

- 1 XML element
- 7 processing instruction
- 8 comment

Example

Here is a set of XML request code.

```
<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <!--this is a comment-->

  <projtitle>

    <projsubtitle>first_subtitle</projsubtitle>

    <projsubtitle>second_subtitle</projsubtitle>

    <projsubtitle>third_subtitle</projsubtitle>

  </projtitle>

</postreq>
```

Here is the PeopleCode that gets types: &xmlprocType is 7 for processing instruction, postreqDoc is 1 for element, and commentType is 8 for comment.

```
Local BIDocs &rootInDoc, &postreqDoc, &commentDoc;

Local number &xmlprocType, &postreqType, &commentDoc;

Local string &blob;

&blob = %Request.GetContentBody();

/* <?xml version="1.0"?> */

&rootInDoc = GetBiDoc(&blob);

&xmlprocType = &rootInDoc.NodeType;
```

```

/* <postreq> */

&postreqDoc = &rootInDoc.GetNode(1);

&postreqType = &postreqDoc.NodeType;

/* <!--this is a comment--> */

&commentDoc = &postreqDoc.GetNode(2);

&commentType = &commentDoc.NodeType;

```

NodeValue

Syntax

NodeValue

Description

The **NodeValue** property gets the value of an XML element referenced by a BiDocs object. This property is read-only.

Parameters

None.

Return Value

String. The node value.

Example

Here is a set of XML request code.

```

<?xml version="1.0"?>

<postreq>

  <email>joe_blow@peoplesoft.com</email>

  <projtitle>

    <projsubtitle>first_subtitle</projsubtitle>

    <projsubtitle>second_subtitle</projsubtitle>

    <projsubtitle>third_subtitle</projsubtitle>

  </projtitle>

```

```
</postreq>
```

Here is the PeopleCode that gets the value of the third `<projsubtitle>` element, `third_subtitle`.

```
Local BiDocs &rootInDoc, &postreqDoc, &projtitleDoc, &projsubtitleDoc;

Local string &name, &blob;

&blob = %Request.GetContentBody();

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetNode(1);

&projtitleDoc = &postreqDoc.GetNode(2);

&projsubtitleDoc = &projtitleDoc.GetNode(3);

&projsubtitleName = &projsubtitleDoc.NodeValue;
```

ParseXMLString

Syntax

```
ParseXMLString (XMLstring)
```

Description

The **ParseXMLString** method fills an existing BiDocs object with the data and shape from an XML string. This is part of the Incoming Business Interlink functionality, which allow PeopleCode to receive an XML request and return an XML response.

Parameters

XMLstring	A string containing an XML document.
-----------	--------------------------------------

Return Values

Number. The return status. NoError, or 0, means the method succeeded.



For more information on the return status, see `GetStatus`.

Example

The following example gets an XML request, creates an empty BiDoc, and then fills the BiDoc with the data and shape contained in the XML string. Once this is done, the `GetDoc` method and

the `GetValue` method can get the value of the skills XML element, which is contained within the `postreq` element in the XML request.

```
Local BiDocs &rootInDoc, &postreqDoc;

Local string &blob;

Local number &ret;

&blob = %Request.GetContentBody();

/* process the incoming xml(request)- Create a BiDoc and fill with the request*/
&rootInDoc = GetBiDoc("");

&ret = &rootInDoc.ParseXMLString(&blob);

&postreqDoc = &rootInDoc.GetDoc("postreq");

&ret = &postreqDoc.GetValue("skills", &skills);
```

Business Interlink and Incoming Business Interlink Functions

This section lists the Business Interlink and Incoming Business Interlink functions.

GetBiDoc

Syntax

`GetBiDoc (XMLstring)`

The **GetBiDoc** function creates a BiDocs object. You can populate the BiDoc with the shape and data from *XMLstring*. This is part of the Incoming Business Interlink functionality, which allow PeopleCode to receive an XML request and return an XML response.

Parameters

<i>XMLstring</i>	A string containing an XML document. You can specify a NULL string for this parameter, that is, two quotation marks ("") without a space between them.
------------------	---

Return Value

BiDocs. The BiDocs structure containing the shape and data of the XML request.

Example

The following example gets an XML request, puts it into a text string, and puts that into a BiDoc. Once this is done, the GetDoc method and the GetValue method can get the value of the skills XML element, which is contained within the postreq XML element in the XML request.

```
Local BIDocs &rootInDoc, &postreqDoc;

Local string &blob;

Local number &ret;

&blob = %Request.GetContentBody();

/* process the incoming xml(request)- Create a BiDoc and fill with the request*/

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetDoc("postreq");

&ret = &postreqDoc.GetValue("skills", &skills);
```

You can also create an empty BiDoc with GetBiDoc, as in the following example.

```
Local BIDocs &rootInDoc, &postreqDoc;

Local string &blob;

Local number &ret;

&blob = %Request.GetContentBody();

/* process the incoming xml(request)- Create a BiDoc and fill with the request*/

&rootInDoc = GetBiDoc("");

&ret = &rootInDoc.ParseXMLString(&blob);

&postreqDoc = &rootInDoc.GetDoc("postreq");

&ret = &postreqDoc.GetValue("skills", &skills);
```

GetInterlink

Syntax

```
GetInterlink(Interlink.interlinkname)
```

Description

The **GetInterlink** function instantiates a Business Interlink Object based on a Business Interlink created in the Application Designer. This function is part of the PeopleSoft Business Interlink framework, which can provide a gateway for PeopleSoft applications to the services of any external system.

After you use this function, you may want to refresh your page. The **Refresh** method, on a rowset object, reloads the rowset (scroll) using the current page keys. This causes the page to be redrawn. `GetLevel0().Refresh()` refreshes the entire page. If you only want a particular scroll to be redrawn, you can refresh just that part.



For more information see the rowset method Refresh.

Parameters

Interlink. <i>interlinkname</i>	Specify the name of the business interlink (created in the Application Designer) from which to instantiate a Business Interlink Object.
--	---

Returns

A Business Interlink Object.

Example

The following example instantiates a Business Interlink Object based on the Business Interlink Definition QE_RP_SRAALL.

```
Local Interlink &SRA_ALL_1;  
  
&SRA_ALL_1 = GetInterlink(Interlink.QE_RP_SRAALL);
```

Related Topics

Business Interlink Object Methods

IScript Functions

This section lists IScript methods.

GetContentBody

Syntax

```
GetContentBody()
```

Description

The **GetContentBody** Request method retrieves the text content of an XML request. This is part of the Incoming Business Interlink functionality, which allow PeopleCode to receive an XML request and return an XML response.

Parameters

None.

Return Value

String. The received XML request.

Example

The following example gets an XML request, puts it into a text string, and puts that into a BiDoc. Once this is done, the GetDoc method and the GetValue method can get the value of the skills XML element, which is contained within the postreq XML element in the XML request.

```
Local BIDocs &rootInDoc, &postreqDoc;

Local string &blob;

Local number &ret;

&blob = %Request.GetContentBody();

/* process the incoming xml(request) - Create a BiDoc */

&rootInDoc = GetBiDoc(&blob);

&postreqDoc = &rootInDoc.GetDoc("postreq");

&ret = &postreqDoc.GetValue("skills", &skills);
```

Interlink Property

StopAtError

Specifies whether the execution of the PeopleCode program will stop when there's an error, or if the PeopleCode program will try to capture the errors. This property takes a Boolean value: True

to halt execution of your PeopleCode program at an error, False to continue executing. The default value is True.

This property is read-write.

Example

```
&QE_RP_SRAALL_1.StopAtError = False;
```

Configuration Parameters

There are two types of configuration parameters: the ones defined by the Business Interlink Plug-in the Business Interlink Definition is based on, and ones that are standard, that is, defined for every Business Interlink Object.

All configuration parameters must be set before you add any data to the input buffers.

Business Interlink Plug-in Configuration Parameters

The configuration parameters defined by the Business Interlink Plug-in are accessed as if they were properties on the Business Interlink Object. That is, in PeopleCode, you assign the value of a configuration parameter by using the Business Interlink Object followed by a dot and the configuration parameter, in this format:

```
&MYINTERLINK.parameter = value;
```

You can also return the value of a configuration parameter:

```
&MYVALUE = &MYINTERLINK.parameter;
```

Each Business Interlink Plug-in has its own set of configuration parameters. For example, a Federal Express Business Interlink uses the configuration parameter URL, while Supply Chain Business Interlinks use SERVER_NAME, RSERVER_HOST, RSERVER_PORT, TIMEOUT, and URL.

You can specify default values for every configuration parameter in the Business Interlink Definition (created in the Application Designer.) These values will be used if you create a PeopleCode template by dragging the Business Interlink Definition from the Project window in the Application Designer to an open PeopleCode editor window.



For more information see Generating the PeopleCode Template.

In the following example, the Business Interlink Object name is QE_RP_SRAALL_1, and the Business Interlink is from Supply Chain:

```
&QE_RP_SRAALL_1.SERVER_NAME = "server";
```

```
&QE_RP_SRAALL_1.RSERVER_HOST = "pt-sun02.peoplesoft.com";  
  
&QE_RP_SRAALL_1.RSERVER_PORT = "9884";  
  
&QE_RP_SRAALL_1.TIMEOUT = 999;  
  
&QE_RP_SRAALL_1.URL = "HTTP://www.PeopleSoft.com";  
  
&QE_RP_SRAALL_1.StopAtError = False;
```

URL Configuration Parameter

Specifies the location and name of the Business Interlink Plug-in to be used to connect to the external system. This configuration parameter takes a string value.

If the plug-in is located on a web server, you have to specify the name of the web server.

If you specify a file, you can specify either a relative or an absolute path:

- If you specify an absolute path, you must specify the drive letter:

```
&MYBI.URL = File://D:\TEMP.MYDLL;
```

- If you specify a relative path, just use the file name:

```
&MYBI.URL = File://TEMP.MYDLL;
```

If you specify a relative path, the system will first look for the file in the Location directory (specified by the user when the Business Interlink was first created), then it will look in the directory where PeopleTools is installed, in the PSTOOLS/Interface Drivers directory.

BIDocValidate Configuration Parameter

When set to ON, a BIDocs object is checked to see if it exists before adding/getting values from it. For example, if the BIDoc object "Rate" does not exist, and BIDocValidate is set to ON, then the following line of code will cause an error:

```
&ret = &biDocs.GetValue("Rate", &RATE);
```

Index

A

- AddAttribute 6-63
- AddComment 6-64
- AddDoc 6-38
- AddInputRow 6-9
- AddNextDoc 6-40
- AddProcessInstruction 6-65
- AddText 6-67
- AddValue 6-43
- Application Engine program
 - generating a Business Interlink PeopleCode template 3-7
 - populating a record with multiple AE programs 6-13, 6-20
- Architecture 1-1
- attribute
 - adding to an XML element 6-63
- AttributeCount 6-68

B

- batch input/output
 - methods to use 6-6
- BI_SEQ_NUM
 - ordering input and output rows with BulkExecute 6-12
 - use with BulkExecute 6-12
 - use with FetchIntoRecord 6-19
- BIDoc
 - adding an input doc 6-38
 - adding input values 6-43
 - adding the next doc to a list 6-40
 - clearing 6-45
 - filling an XML string with 6-72
 - filling with an XML string 6-80, 6-81
 - getting an output doc 6-49
 - getting output values 6-57
 - getting the next doc in a list 6-51
 - getting the number of docs 6-47
 - getting the previous doc in a list 6-54
 - getting the root BIDoc input doc 6-31
 - getting the root BIDoc output doc 6-32
 - moving to a doc in an output doc list 6-59
 - resetting to the first output doc 6-61
 - status of a method 6-56
- BIDocs
 - getting the number of outputs with GetFieldCount 6-26

- getting the output type with GetFieldType 6-28
- getting the output values with GetFieldType 6-30
- methods used with input 6-2
- methods used with output 6-4
- BIDocs methods
 - using 6-2
- BIDocValidate 6-86
- BIDocValidate configuration parameter 2-10
- BulkExec ID 2-6
- BulkExecute 6-11
 - BI_SEQ_NUM 6-12
 - BulkExec ID 2-6
 - OPERID 6-13
 - PROCESS_INSTANCE 6-13
- Business Interlink
 - actions when running 1-2
 - instantiating 6-83
 - methods 6-9, 6-38
- Business Interlink Architecture 1-1
- Business Interlink Definition
 - designing 2-1
 - saving 2-4
 - selecting 2-3
 - selecting inputs, outputs, data members 2-15
 - setting the parameters 2-4
- Business Interlink methods
 - see methods 6-1
- Business Interlink Object
 - actions taken to create 1-4
 - declaring in PeopleCode 6-8
 - executing 6-17
 - scope in PeopleCode 6-8
 - state 6-8
- Business Interlink page
 - adding a row to the grid 2-15
 - configuration parameters 2-10
 - criteria 2-6
 - displaying data type 2-15
 - displaying required status 2-15
 - expanding transaction, class, or object instances 2-15
 - input 2-5
 - logical operator 2-7
 - operator 2-7
 - output 2-8
 - relational operator 2-7
 - removing a row from the grid 2-15
 - restrictions on dragging from control tree to grid 2-16
 - selecting inputs, outputs, data members 2-15
 - using the tree control and grid 2-15
- Business Interlink PeopleCode template

- dragging a Business Interlink into PeopleCode
 - 3-6, 3-9
- generating in an Application Engine program 3-7
- generating in an event 3-1
- Business Interlink Search page
 - selecting from 2-3
- Business Interlink Tester
 - entering input values 2-12
 - entering multiple sets of input values 2-13
 - printing 2-14
 - saving input values 2-13
 - using 2-11
 - viewing multiple sets of output values 2-14
 - viewing output values 2-14

C

- CD-ROM
 - ordering ii
- child node
 - counting in an XML request 6-69
- ChildNodeCount 6-69
- Clear 6-16, 6-45
 - use with Execute 6-17
- clients
 - populating a record from multiple clients 6-13, 6-20
- comment
 - adding an XML comment 6-64
- configuration parameters
 - accessing 6-85
 - adding to a Business Interlink Definition 2-10
 - BIDocValidate 6-86
 - Business Interlink page 2-10
 - setting defaults 2-10
 - setting for Business Interlinks 2-10
 - URL 6-86
- CreateElement 6-70
- criteria
 - setting for Business Interlinks 2-6

D

- data member
 - adding to a Business Interlink Definition 2-7
 - removing from Business Interlink Definition 2-15
 - selecting for a Business Interlink Definition 2-15
- data type
 - displaying 2-15
- dynamic output
 - getting next row of outputs 6-37
 - getting output type 6-28
 - getting output values 6-30
 - getting the number of outputs 6-26
 - methods to use 6-6
 - moving to top of output buffer 6-36

E

- error
 - stopping PeopleCode execution on 6-84
- event
 - generating a Business Interlink PeopleCode template 3-1
- Execute 6-17
 - use with Clear 6-17
 - use with StopAtError 6-17

F

- FetchIntoRecord 6-19
 - OPERID 6-20
 - PROCESS_INSTANCE 6-20
- FetchIntoRowset 6-22
- FetchNextRow 6-25
- flat table input/output
 - adding a row of input 6-9
 - fetching a row of output 6-25
 - methods to use 6-6

G

- GenXMLString 6-72
- GetAttributeName 6-73
- GetAttributeValue 6-74
- GetBiDoc 6-81
- GetContentBody 6-84
- GetCount 6-47
- GetDoc 6-49
- GetFieldCount 6-26
- GetFieldType 6-28
- GetFieldValue 6-30
- GetInputDocs 6-31
- GetInterlink 6-83
- GetNextDoc 6-51
- GetNode 6-75
- GetOutputDocs 6-32
- GetPreviousDoc 6-54
- GetStatus 6-56
- GetValue 6-57
- grid 2-15
 - adding a row 2-15
 - removing a row 2-15
 - restrictions on adding 2-16
- group
 - query 2-7

H

- hierarchical data structures
 - preserving 6-22, 6-33

I

Incoming Business Interlink

- actions taken to create 1-5
- design-time plug-in for testing 5-12
- filling a BiDoc with an XML string 6-80, 6-81
- filling an XML String with a BiDoc 6-72
- methods to use 6-7
- retrieving the text content of a request body 6-84
- setting up for writing the IScript function 5-1
- testing 5-4
- writing the IScript fuction 5-5
- XML request example 5-9
- XML response example 5-10

input

- adding a row of flat table input 6-9
- clearing the buffer 6-16
- ordering with BulkExecute 6-12
- removing from Business Interlink Definition 2-15

input doc

- adding 6-38
- adding the next doc to a list 6-40
- adding values 6-43
- getting the root BIDoc input doc 6-31

input docs

- entering values for testing 2-13
- methods used with 6-2

input name 2-5

input parameters

- entering multiple input sets for testing 2-13
- entering values for testing 2-12
- name used inside PeopleCode 2-5
- saving values for testing 2-13
- selecting for a Business Interlink Definition 2-5
- setting defaults 2-5
- setting for Business Interlinks 2-5

InputRowset 6-33

IScript function

- setting up for Incoming Business Interlink 5-1
- testing for Incoming Business Interlink 5-4
- writing for Incoming Business Interlink 5-5

L

lists

- restrictions on adding to Business Interlink Definition 2-16

logical

- Business Interlink page 2-7

M

method

- AddAttribute 6-63

- AddComment 6-64
- AddDoc 6-38
- AddInputRow 6-9
- AddNextDoc 6-40
- AddProcessInstruction 6-65
- AddText 6-67
- AddValue 6-43
- BulkExecute 6-11
- Clear 6-16, 6-45
- CreateElement 6-70
- Execute 6-17
- FetchIntoRecord 6-19
- FetchIntoRowset 6-22
- FetchNextRow 6-25
- GenXMLString 6-72
- GetAttributeName 6-73
- GetAttributeValue 6-74
- GetBiDoc 6-81
- GetContentBody 6-84
- GetCount 6-47
- GetDoc 6-49
- GetFieldCount 6-26
- GetFieldType 6-28
- GetFieldValue 6-30
- GetInputDocs 6-31
- GetNextDoc 6-51
- GetNode 6-75
- GetOutputDocs 6-32
- GetPreviousDoc 6-54
- GetStatus 6-56
- GetValue 6-57
- InputRowset 6-33
- MoveFirst 6-36
- MoveNext 6-37
- MoveToDoc 6-59
- ParseXMLString 6-80
- ResetCursor 6-61

methods

- batch input/output 6-6
- deciding which methods to use 6-1
- dynamic output 6-6
- flat table input/output 6-6
- Incoming Business Interlink 6-7
- list of 6-9, 6-38
- methods used for standard input/output 6-2
- rowsets 6-6
- MoveFirst 6-36
- MoveNext 6-37
- MoveToDoc 6-59

N

name

- getting for an XML element 6-76

New Business Interlink dialog box 2-2

node

- getting the name of an attribute from an XML element 6-73
- getting the number of attributes for an XML element 6-68
- NodeName 6-76
- NodeType 6-77
- NodeValue 6-79
- NOT
 - query 2-7

O

- operator
 - Business Interlink page 2-7
- OPERID
 - FetchIntoRecord 6-20
 - use with BulkExecute 6-13
- order by 2-8
- output
 - clearing the buffer 6-16
 - dynamic
 - see dynamic output 6-26
 - fetching a row of flat table output 6-25
 - ordering with BulkExecute 6-12
 - ordering with FetchIntoRecord 6-19
 - removing from Business Interlink Definition 2-15
 - writing large amount of data to a record 6-19
- output doc
 - adding 6-49
 - clearing 6-45
 - getting number of docs 6-47
 - getting the next doc in a list 6-51
 - getting the previous doc in a list 6-54
 - getting the root BIDoc output doc 6-32
 - getting values 6-57
 - moving to a doc in a list 6-59
 - resetting to the first 6-61
- output docs
 - methods used with 6-4
 - viewing values for testing 2-14
- output name 2-8
- output parameters
 - adding to a Business Interlink Definition 2-8
 - name used inside PeopleCode 2-8
 - order by for a query 2-8
 - selecting for a Business Interlink Definition 2-15
 - setting for Business Interlinks 2-8
 - viewing multiple output sets for testing 2-14
 - viewing values from testing 2-14

P

- ParseXMLString 6-80
- PeopleBooks
 - CD-ROM, ordering ii

- printed, ordering iii
- PeopleCode
 - generating a Business Interlink template in an event 3-1
 - stopping execution on error 6-84
- PeopleCode methods
 - see methods 6-1
- PeopleCode template
 - example for query 4-13
 - example for transaction 4-3
 - example of fully coded 4-6
 - filling in 4-1
 - main sections 4-2
 - see Business Interlink PeopleCode Template 3-1
- printing
 - Business Interlink Tester 2-14
- PROCESS_INSTANCE
 - use with BulkExecute 6-13
 - use with FetchIntoRecord 6-20
- property
 - AttributeCount 6-68
 - ChildNodeCount 6-69
 - NodeName 6-76
 - NodeType 6-77
 - NodeValue 6-79

Q

- query
 - ascending or descending output order 2-9
 - NOT 2-7
 - ordering outputs 2-8
 - right-hand-side 2-7
 - tabs to set Business Interlink Definition 2-4
 - ungroup 2-7

R

- records
 - populating from multiple clients 6-13, 6-20
 - populating with multiple Application Engine programs 6-13, 6-20
 - using with BulkExecute 6-11
 - writing large amounts of data 6-19
- required status
 - displaying 2-15
- ResetCursor 6-61
- rowset
 - reading from rowset with InputRowset 6-33
 - writing to rowset with FetchIntoRowset 6-22
- rowsets
 - methods to use 6-6

S

- scope
 - Business Interlink Object 6-8
- searching for Business Interlink plug-in 2-2
- settings
 - Business Interlink page 2-10
- state
 - Business Interlink Object 6-8
- status
 - getting for BIDoc methods 6-56
- StopAtError 6-84
 - use with Execute 6-17
- Supply Chain
 - restrictions on adding to Business Interlink Definition 2-16

T

- testing Business Interlink Definitions 2-11
- text
 - adding to an XML element 6-67
- transaction
 - processing large amounts of data 6-11
 - tabs to set Business Interlink Definition 2-4
- Transaction Search list box 2-3
- tree control 2-15
 - expanding transaction, class, or object instances 2-15
 - restrictions on adding to grid 2-16
- type
 - getting for an XML element 6-77

U

- ungroup

- query 2-7
- URL 6-86
- URL configuration parameter 2-10

V

- value
 - getting for an attribute in an XML element 6-74
 - getting for an XML element 6-79

X

- XML comment
 - adding a comment to a BiDocs object 6-64
- XML doc
 - getting an XML element 6-75
- XML element
 - adding an attribute 6-63
 - adding text 6-67
 - adding to a BiDoc object 6-70
 - getting from a BiDocs object 6-75
 - getting the name 6-76
 - getting the name of an attribute 6-73
 - getting the number of attributes 6-68
 - getting the number of child nodes 6-69
 - getting the type 6-77
 - getting the value 6-79
 - getting the value of an attribute 6-74
- XML processing instruction
 - adding to an XML request 6-65
- XML request
 - retrieving text 6-84
- XML request example for Incoming Business Interlink 5-9
- XML response example for Incoming Business Interlink 5-10

