ORACLE
PEOPLESOFT

Oracle's PeopleTools PeopleBook

PeopleTools 8.52: Supported Integration Technologies

June 2013

ORACLE

PeopleTools 8.52: Supported Integration Technologies
SKU pt8.52tsis-b0613

Contents

Supported Integration Technologies Preface

This preface provides an overview of supported integration technologies.

Supported Integration Technologies

Before PeopleSoft 8, PeopleSoft Tools & Technology (PeopleTools) delivered a variety of tools and technologies for integrating PeopleSoft applications with each other and with third-party products and services. Typically, these technologies were narrowly focused on industry-specific transactions and data formats. Although we recommend the more robust and adaptable tools delivered with the current release, the older technologies documented in this PeopleBook are still supported.

PeopleBooks and the PeopleSoft Online Library

A companion PeopleBook called *PeopleBooks and the PeopleSoft Online Library* contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.
- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.
- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the locally installed PeopleSoft online library, including web site folders.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Application abbreviations found in application fields.

You can find *PeopleBooks and the PeopleSoft Online Library* in the online PeopleBooks Library for your PeopleTools release.

Chapter 1

Getting Started with Supported Integration Technologies

This chapter provides an overview of Supported Integration Technologies and discusses:

- Supported integration technologies implementation.
- Other sources of information.

Supported Integration Technologies Overview

This PeopleBook describes several integration technologies that PeopleSoft delivered before PeopleSoft 8.

Note. Oracle supports the older technologies documented in this PeopleBook; however, you should use the more robust and adaptable tools delivered with the current release.

EDI Manager	Electronic Data Interchange (EDI) is a standard means of exchanging data between companies so that they can transact business electronically. PeopleSoft EDI Manager enables you to define event and action codes, define EDI transactions, set up trading partners, map transactions, and monitor transaction processing.
Outgoing Forms API	Forms routing enables the system to take data from a PeopleSoft application page on which a user is working, enter it into a third-party form, and mail the completed form to designated users by means of the forms product's mail capabilities. The PeopleSoft system provides a PSFORMS dynamic link library (PSFORMS.DLL) that PeopleSoft applications use to communicate with forms software, which includes session-level, query, and send operations.
Open Query ODBC Driver and API	The PeopleSoft Open Query ODBC driver and API enable third-party reporting tools or applications to access PeopleSoft data in conformance with the PeopleSoft Query access architecture (the embedded SQL access intelligence provided by PeopleSoft Query).

Supported Integration Technologies Implementation

The technologies discussed in this PeopleBook are automatically installed with PeopleTools. In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides and release notes. Moreover, in addition to the information in this PeopleBook, refer to the following PeopleBooks:

Technology	***PeopleBook***
EDI Manager	*PeopleTools 8.52: SQR for PeopleSoft Developers PeopleBook*
Outgoing Forms API	*PeopleTools 8.52: PeopleSoft Application Designer Developer's Guide PeopleBook*
Open Query ODBC Driver and API	*PeopleTools 8.52: PeopleSoft Query PeopleBook* and *PeopleTools 8.52: Crystal Reports for PeopleSoft PeopleBook*

Other Sources of Information

In addition to the implementation considerations presented in this section, take advantage of all sources of information about PeopleSoft products, including the installation guides, release notes, PeopleBooks, and curriculum.

See Also

"Supported Integration Technologies Preface," page vii

PeopleTools 8.52: Getting Started with PeopleTools

Chapter 2

Using PeopleSoft EDI Manager

This chapter provides an overview of Electronic Data Interchange (EDI) and discusses how to:

- Define event and action codes.
- Specify data conversion values.
- Define EDI transactions.
- Set up trading partners.
- Delete PeopleSoft EDI Manager objects.

Note. EDI Manager should no longer be used to load or unload flat files that are not X.12 or EDIFACT EDI transactions. File layouts have stronger file access methods, with batch file import and export, interactive import data troubleshooting, and automatic generation of import PeopleCode.

Understanding EDI

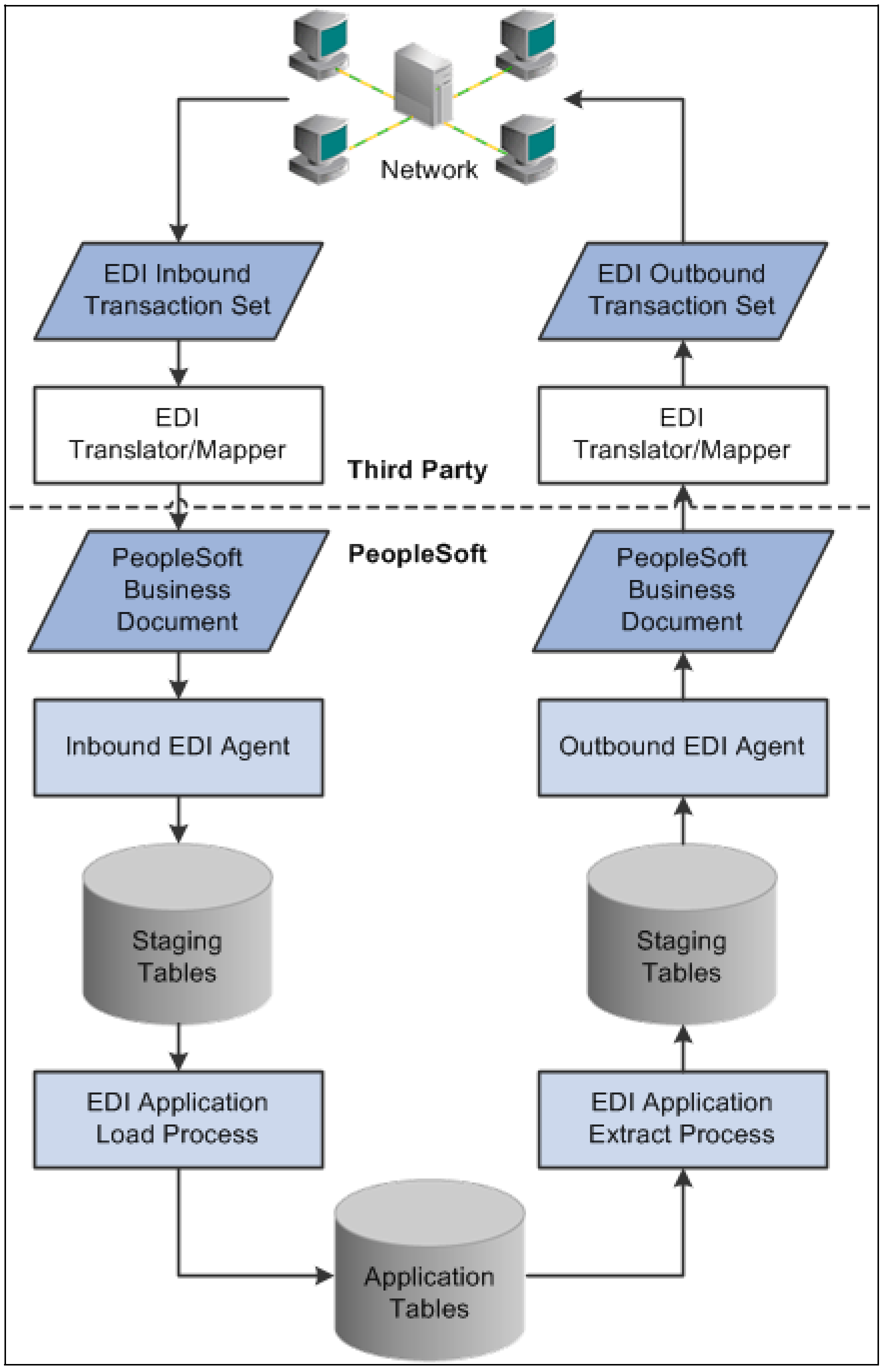
This section provides an overview of EDI and discusses EDI codes and PeopleSoft codes.

EDI Overview

EDI is a standard means of exchanging data between companies so that they can transact business electronically. For example, using EDI, a company can submit an order to a vendor, and the vendor can acknowledge and fulfill the order without any paper changing hands or any contact between company representatives.

EDI provides a standard format for transaction data, enabling trading partners to communicate in a common language. When one company needs to initiate a transaction with another, it extracts the transaction data from its database, translates it into the common EDI format, and transmits it over a network to the trading partner. The second company receives the EDI transmission and transfers its data into its transaction processing application.

The EDI process involves several steps. This diagram shows an overview of the architecture that enables PeopleSoft applications to complete EDI transactions:


Network
EDI Inbound Transaction Set
EDI Outbound Transaction Set
EDI Translator/Mapper
EDI Translator/Mapper
Third Party
PeopleSoft
PeopleSoft Business Document
PeopleSoft Business Document
Inbound EDI Agent
Outbound EDI Agent
Staging Tables
Staging Tables
EDI Application Load Process
EDI Application Extract Process
Application Tables

PeopleSoft EDI architecture

The portion above the dotted line shows the services typically provided by a value-added network (VAN). A VAN provides services related to the exchange of EDI transactions—usually a private network for exchanging EDI transactions, although the network could also be the internet.

For incoming transactions, the transfer to PeopleSoft applications comes when the EDI translation software converts transactions from the standard EDI formats X.12 or EDIFACT into PeopleSoft Business Document format. A PeopleSoft-supplied EDI agent reads the PeopleSoft Business Document files and places their data in staging tables in the PeopleSoft database. An EDI application load Structured Query Report (SQR) transfers data from the staging tables to the appropriate application tables. From there, the data is processed like any other PeopleSoft transaction.

The process for outgoing transactions is similar. You periodically run an SQR that extracts the appropriate data from the application tables and copies it to a set of staging tables. An EDI agent creates PeopleSoft business documents from this data; then the EDI translation software converts the documents into X.12 or EDIFACT format and transmits them to the trading partner.

The portion of the process involving PeopleSoft products has two stages:

- Based on data mappings that you define, EDI agents copy data between PeopleSoft business documents and the incoming or outgoing staging tables.
- SQRs transfer data between the staging tables and the application tables, performing any necessary data validation.

You use PeopleSoft EDI Manager to define the mappings that EDI agents use for conversions and to maintain information about your vendors and other trading partners.

The staging tables are a temporary storage area for EDI transaction data. Later, a transaction-specific application load process transfers the data from the staging tables into the application tables.

EDI Codes and PeopleSoft Codes

When you receive an EDI transaction from a trading partner, much of its data is in the form of codes or identification numbers. For example, the trading partner who submitted the transaction is given as a trading partner ID, and the type of transaction is identified by a transaction ID.

Your PeopleSoft database also stores much of its data in the form of codes or IDs. You have business unit IDs, customer IDs, employee IDs, user IDs, and so on. Some of these codes and IDs represent the same data as the codes and IDs in the EDI transaction, even though the codes themselves might be different.

When the EDI agent processes a transaction, it copies the transaction data into the PeopleSoft database. As it does so, it can convert the external EDI codes into the internal PeopleSoft codes. Similarly, as it writes out the file for an outgoing EDI transaction, it can convert the PeopleSoft codes into codes that your trading partner will recognize. You specify how the EDI agent converts the codes using PeopleSoft EDI Manager.

PeopleSoft EDI Manager can perform two kinds of conversions:

- It can translate between EDI event codes and PeopleSoft action codes, which specify what action a transaction requires.
- It can convert data values from any field in the transaction.

Defining Event Codes and Action Codes

This section provides an overview of event codes and action codes and discusses how to:

- Define event codes.
- Define action codes.

Understanding Event Codes and Action Codes

EDI transactions use event codes to specify what action the transaction calls for. For example, event codes specify whether the current transaction is a completely new transaction, a resubmitted transaction, or an update to a previous transaction.

EDI event codes come in two types, both of which can be specified for the same transaction:

- Primary event codes, also called purpose codes, specify the status of the transaction: whether it is a new transaction, a cancellation, a duplicate, a status request, and so on. Every transaction has a primary event code.
- Secondary event codes, also called transaction codes, specify the type of transaction in detail. For example, a transaction's secondary event code could say that the transaction is a catalog order, a rush order, or a request for a sample. Not all transaction types have secondary event codes.

PeopleSoft applications, on the other hand, determine the action to take on a transaction using a single action code. Therefore, when an EDI agent processes an EDI transaction, it needs to convert its event codes into a PeopleSoft action code.

Using PeopleSoft EDI Manager, you specify which pairs of event codes are translated into which action codes. Because you may want to process transactions differently, depending on which trading partner they come from, EDI Manager enables you to define different event code-to-action code translations for each trading partner.

After you complete the following activities, you must specify a profile for each trading partner.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," Setting Up Trading Partners, page 16

Pages Used to Define Event Codes and Action Codes

Page Name	*Definition Name*	*Navigation*	*Usage*
EC Primary Event Table	EC_PRI_EVENT_TABLE	PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Primary Event Codes	Define primary event codes.
EC Sec Event Table	EC_SEC_EVENT_TABLE	PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Secondary Event Codes	Define secondary event codes.
EC Action Code Definition	EC_ACTION_CODES	PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Action Codes	Define action codes.

Defining Event Codes

To define primary or secondary event codes:

1. Select PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code.
2. Select Primary Event Codes or Secondary Event Codes.

   You can search for an existing value or add a new value by clicking the link.
3. Enter the event code as it will appear in PeopleSoft business documents.
4. Enter a description of the event code and save the page.

   This description is for your information only. It does not affect processing.

Defining Action Codes

PeopleSoft delivers EDI Manager with a complete set of codes for the actions that standard PeopleSoft applications support. Add new action codes only if you create new programs that handle new actions.

To define an action code:

1. Select PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Action Codes.
2. Click the Add a New Value tab.
3. In the EC Action Code field, enter the action code as you want the EDI agent to write it in the electronic commerce staging tables.

   Click the Add button.

   The EC Action Code Definition page appears.
4. Enter a description of the action code and save the page.

   This description is for your information only. It does not affect processing.

Specifying Data Conversion Values

This section provides an overview of specifying data conversion values and discusses how to:

- Assign trading partner conversion IDs.
- Create conversion data profiles.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," Setting Up Trading Partners, page 16

Understanding Specifying Data Conversion Values

In many cases, the set of possible values in a particular field of an EDI transaction does not match the corresponding set of values in the PeopleSoft database. For example, the EDI transaction might identify bank transaction codes using three-digit numbers, while the PeopleSoft database uses single letters to represent the same codes.

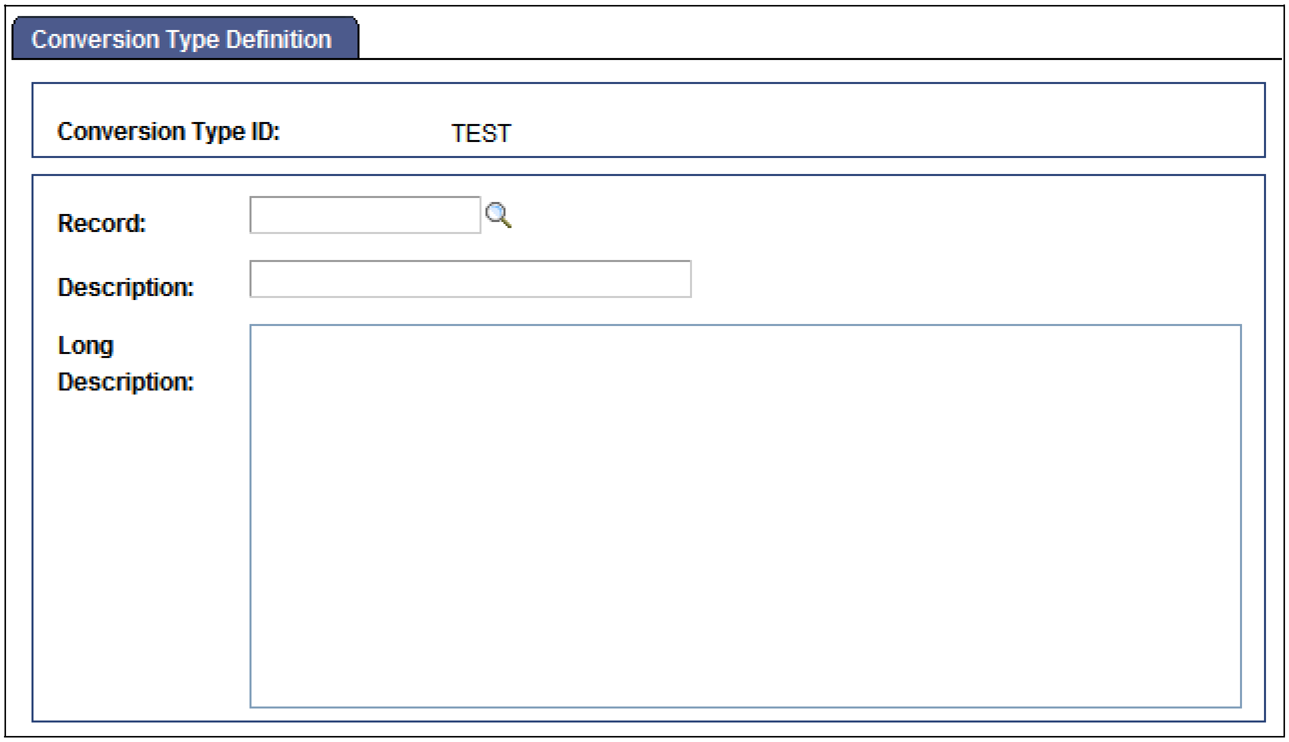
In such cases, PeopleSoft EDI Manager needs to translate the external values into the corresponding internal values.

Pages Used to Specify Data Conversion Values

Page Name	***Definition Name***	***Navigation***	***Usage***
Conversion Type Definition	EC_ECTPCVT	PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Conversion Types	Identify database tables whose values need to be converted.
Conversion Data Profile	EC_ECTPCVT_VALUES	PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Conversion Data Profile	Specify how values from a PeopleSoft database table appear in PeopleSoft business documents.

Assigning Trading Partner Conversion IDs

Access the Conversion Type Definition page.



Conversion Type ID:	TEST
Record:	
Description:	
Long Description:	

Conversion Type Definition page

To identify a database table whose values need to be converted:

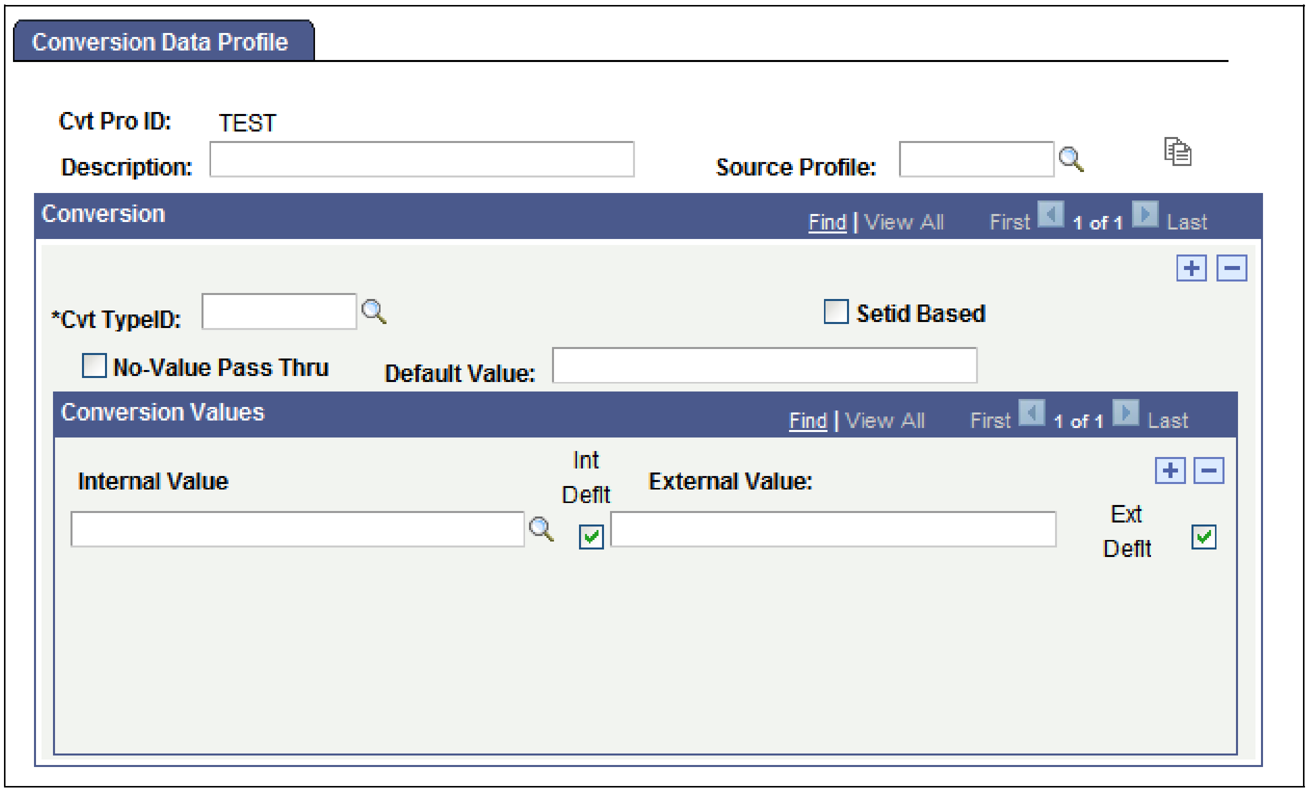
1. Select PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Conversion Types.
2. Enter a conversion type ID or search for an existing one.

   You can also search for the record name.
3. Enter the record name of the table whose values you want to convert and enter a description.

   You can enter a description in the Description text box, the large edit box, or both.
4. Save the page.

Creating Conversion Data Profiles

Access the Conversion Data Profile page.


Conversion Data Profile

Cvt Pro ID: TEST

Description:

Source Profile:

Conversion — Find | View All — First 1 of 1 Last

*Cvt TypeID:

Setid Based

No-Value Pass Thru

Default Value:

Conversion Values — Find | View All — First 1 of 1 Last

Internal Value	Int Deflt	External Value:	Ext Deflt
	✓		✓

Conversion Data Profile page

A conversion data profile takes the values from a particular PeopleSoft database table (such as the table holding bank transaction codes) and specifies how the values appear in PeopleSoft business documents. You can create multiple conversion data profiles for the same table, because you might need to create different conversions for different trading partners. For example, different banks might use different transaction codes.

To create a conversion data profile:

1. Select PeopleTools, EDI Manager, Convert EDI/PeopleSoft Code, Conversion Data Profile.
2. Search for an existing Conversion data profile or enter a new value.

   The Conversion Data Profile page appears. You use this page to specify translation values for one or more database tables, identified here by their conversion profile IDs.

   The page includes two areas to enter data. The outer area enables you to add multiple conversion type IDs to the conversion profile. The inner area enables you to scroll among the values in the table identified by the conversion ID.
3. (Optional.) Copy the definition of a similar conversion data profile.

   If the profile you are creating is similar to an existing profile, you can copy the details of the existing profile definition into this component. Then, you can change just the parts that are different.

   Select the existing Source Profile, and then click the EC Copy Conversion Profile button. The system copies the conversion values from the selected profile.

4. Enter a description of the conversion data profile.

   You can enter both a short description (in the Description text box) and a long description (in the long edit box). These descriptions are for your information only; the system does not use them.

5. Select a conversion type ID.

   In the Cvt TypeID field, select the trading partner conversion type ID of the table for which you want to provide external values.

6. Specify whether the table whose values you are converting has a setID key.

   Many tables in PeopleSoft Financials applications have a setID key. The setID key enables you to store data for multiple business units on the same table while making sure that each unit only accesses its own data. Selecting a setID controls the list of internal values available for mapping: if the table you are retrieving the internal values from has a SetID field, then the Conversion Data Profile page only displays the values for the selected setID.

   To select a setID, select the SetID Based check box, and then select the appropriate *SetID* in the field that appears.

7. Select an internal value and enter the corresponding external value.

   In the Internal Value drop-down list, select a value from the table whose conversion ID you selected. Then, in the External Value text box, type the corresponding value as it will appear in PeopleSoft business documents.

8. Specify which internal and external values to use when multiple possibilities exist.

   One potential complication of having many-to-one mapping occurs with outbound transactions: for a given internal value, what value should the outbound EDI agent write into the outbound file? In this situation, the outbound EDI agent uses the row marked as the internal default (Int Deflt field) value.

   The same scenario holds in reverse when you have multiple internal database values that you need to map to the same external value. No problem occurs with outbound transactions: the outbound EDI agent maps the internal value to the appropriate external value. For incoming transactions, however, you need to specify which of the multiple internal values to use. You do so by selecting one of the rows as the external default (Ext Defltfield) value.

   When you need to convert multiple external values to the same internal value, you should mark each of the external values as an external default, and then mark one of the rows as the internal default. When you need to convert multiple internal values to the same external value, mark each of the internal values as an internal default, and then mark one row as the external default.

   Note. You must perform this step even if all mappings are one-to-one mappings. Select the Int Deflt and Ext Deflt check boxes for every row.

9. Repeat the previous steps to add additional conversion IDs.

Defining EDI Transactions

This section provides an overview of defining EDI transactions and discusses how to:

- Create transaction IDs.

- Define partner profiles.
- Set profile defaults.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," Setting Up Trading Partners, page 16

Understanding Defining EDI Transactions

When you receive an EDI transaction from a trading partner, the first record of the transaction includes a transaction ID that identifies the transaction type. The EDI agent uses the transaction ID, in conjunction with the trading partner ID, to determine which inbound map to use to process the transaction data. Similarly, when you initiate an outbound transaction, the EDI agent puts the appropriate transaction ID in the first transaction record so that the recipient knows what kind of transaction you have sent.

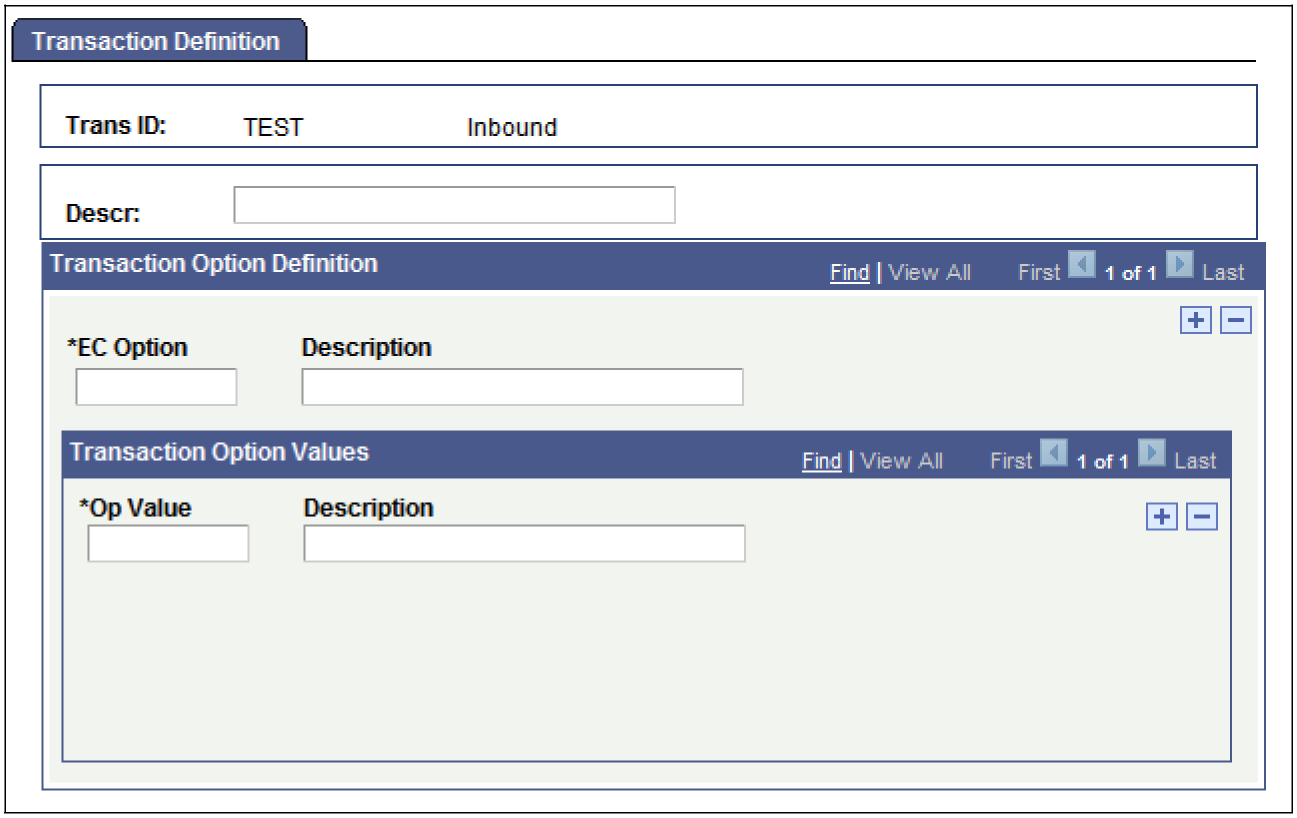
In PeopleSoft EDI Manager, you must identify the transactions that your system is prepared to process. You also must specify, for each of your trading partners, the transactions that they are authorized to perform. After you complete these steps, you must assign the profiles to your trading partners.

Pages Used to Define EDI Transactions

Page Name	***Definition Name***	***Navigation***	***Usage***
Transaction Definition	EC_TRANS_PNL	PeopleTools, EDI Manager, Define EDI Transactions, Transactions	Create EDI transaction IDs.
Partner Profile - Profile Definition	EC_TP_PROFILE_1	PeopleTools, EDI Manager, Define EDI Transactions, Partner Profile Click the Profile Definition tab.	Define partner profiles.
Partner Profile - Profile Defaults	EC_TP_PROFILE_2	PeopleTools, EDI Manager, Define EDI Transactions, Partner Profile Click the Profile Defaults tab.	Specify the transactions a partner with a specified profile can perform and how the EDI agent converts event codes into action codes for each transaction.

Creating Transaction IDs

Access the Transaction Definition page.


Transaction Definition
Find | View All First 1 of 1 Last
Find | View All First 1 of 1 Last

Trans ID:	TEST	Inbound
Descr:		

Transaction Option Definition

*EC Option	Description

Transaction Option Values

*Op Value	Description

Transaction Definition page

To define a transaction:

1. Add a new value and enter a unique EC transaction ID as it will appear in PeopleSoft business documents.
2. Specify whether you are setting up an ID for inbound or outbound transactions.

   Note. If you support both inbound and outbound transactions of the same transaction type, you must define two transaction IDs.

3. Click the Add button.

   The Transaction Definition page appears. You use this page to define the transaction and specify any special transaction parameters.

4. Enter a description of the transaction in the Descr (description) text box.

   The description appears in list boxes when you need to select a transaction ID.

5. Enter the name of a transaction option that the application load or application extract process uses.

   Use the EC Option text box to specify a transaction parameter that the system will use when it copies data from the electronic commerce staging tables to the transactional application tables.

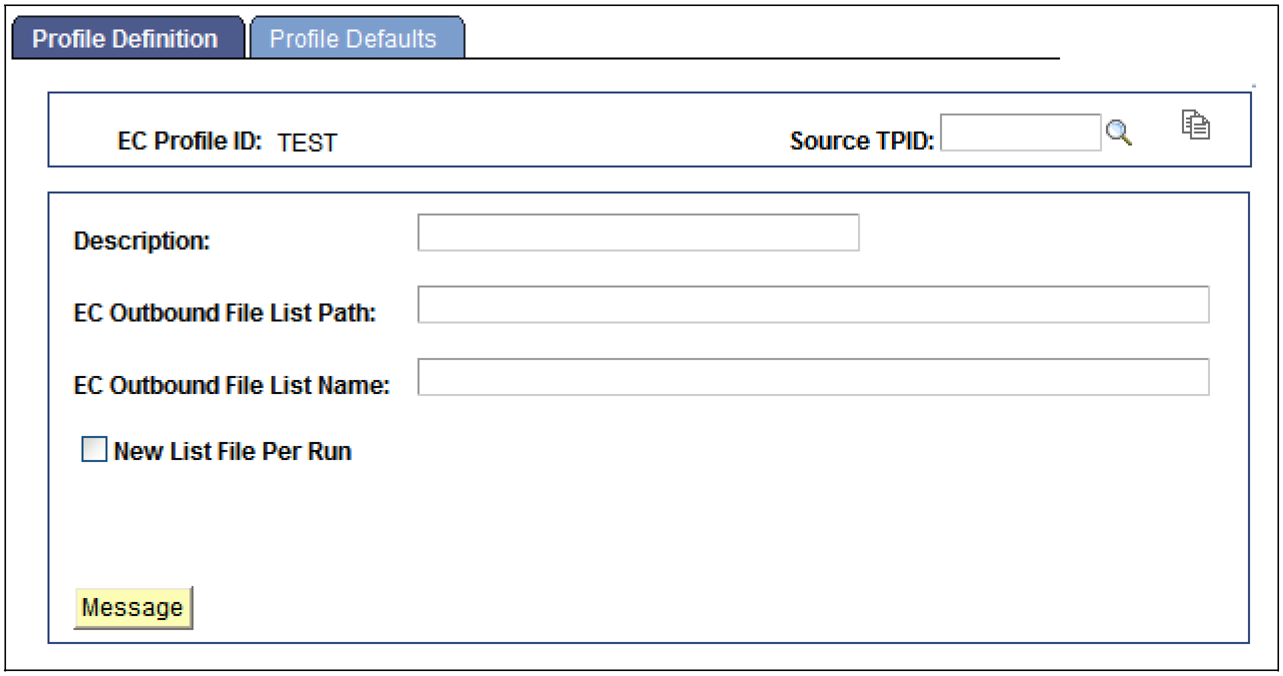
6. Enter the valid transaction option values for the transaction option.

   List the valid values in the Op Value field for the transaction option that you specified in the previous step. Add a row for each option value. When you define a trading partner, you will specify which of these values to use for that partner. In this way, the application load or application extract process can handle the transaction differently depending on the trading partner that it comes from.

   Repeat steps 5 and 6 to define additional transaction options.

Defining Partner Profiles

Access the Partner Profile - Profile Definition page.



EC Profile ID: TEST

Source TPID:

Description:

EC Outbound File List Path:

EC Outbound File List Name:

☐ New List File Per Run

Partner Profile - Profile Definition page

To define a partner profile:

1. (Optional) Copy the definition of a similar partner profile.

   If the profile that you are creating is similar to an existing profile, you can copy the details of the existing profile definition into this component. Then, you can change just the parts that are different.

   Select the existing Source TPID, and then click the EC Copy Profile Function button. The system copies the values from the selected profile.

2. Specify the directory where you want the EDI agent to write outbound transaction files for partners with this profile.

   Enter the directory path in the EC Outbound File List Path text box.

   Note. If PeopleSoft EDI Manager runs on a server under MVS, it ignores this setting. Instead, it looks in the JCL for a DDNAME matching the file name that you specify in the next step.

   ---

3. Enter a name for the file that lists all pending outbound transaction files.

   As the EDI agent creates transaction files, it adds the names of those files to a list file, the name of which you enter in the EC Outbound File List Name text box.

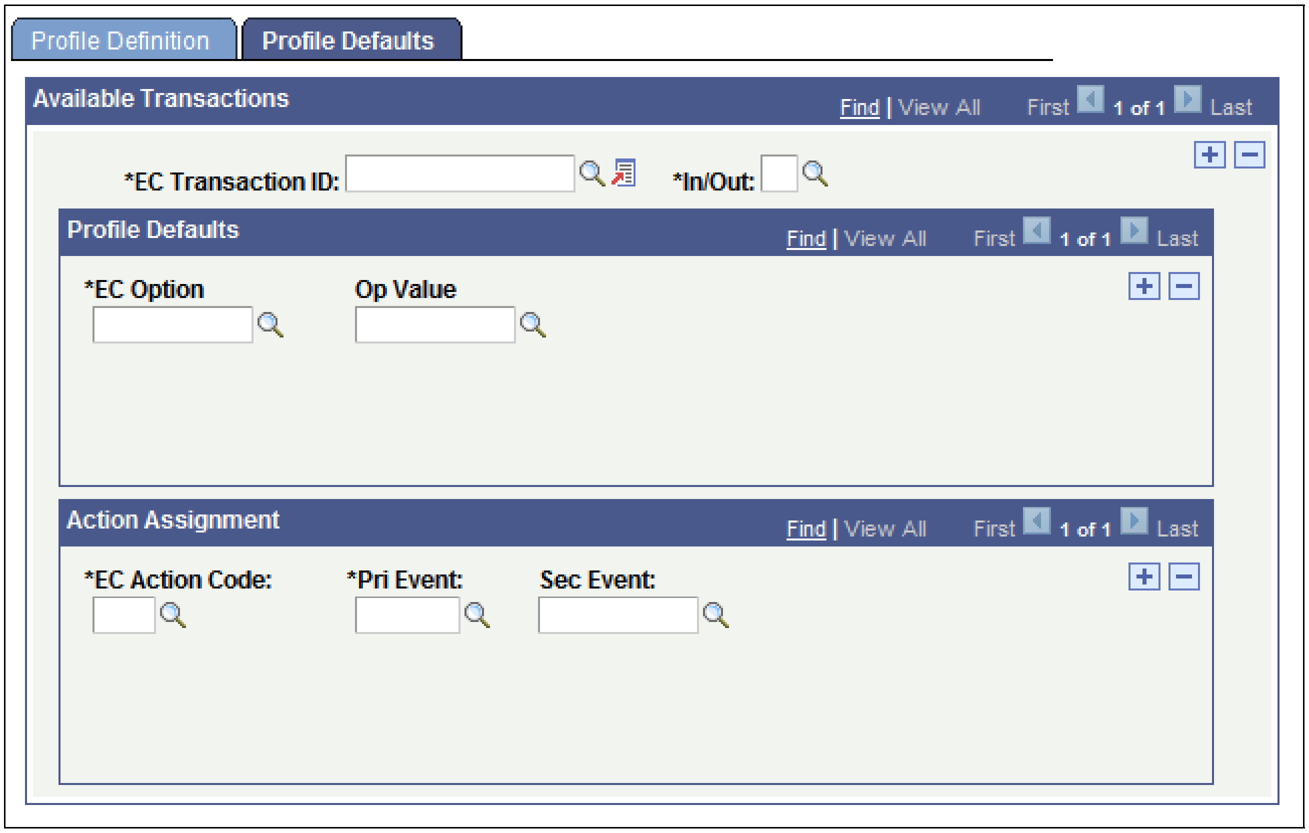
   The transaction files themselves are automatically named 1.DAT, 2.DAT, and so on, except on MVS servers. On MVS, the files are named using the first eight characters of the external trading partner ID.

4. (Optional) Select New List File Per Run if you want a new list generated every run.

5. (Optional) Click the Message button to insert a standard message header or footer in the file.

Setting Profile Defaults

Access the Partner Profile - Profile Defaults page.



Partner Profile - Profile Defaults page

Use this page to specify which transactions a partner with this profile can perform and how the EDI agent converts event codes into action codes for each transaction. If a transaction has transaction options associated with it, you also specify which value to assign to that option for partners with this profile.

To set profile defaults:

1. Select a transaction type that partners with this profile can perform.

   Select the transaction ID in the EC Transaction ID field. Specify whether it is an inbound or outbound transaction by selecting *I* or *O* in the In/Outfield.

2. If the transaction type has transaction options associated with it, specify which value to use for each one in the Profile Defaults group box.

   When you define a transaction type, you can specify transaction options that the system uses when it copies data between the electronic commerce staging tables into the transactional application tables. If the transaction type that you are adding includes options, specify which of the available values for that option the EDI agent should use for partners with this profile.

   Note. Most transaction types do not have any options, and you can bypass this step.

   Select the option in the EC Option field. Then, select one of its valid values in the Op Value field. Repeat this step for each defined transaction option.

3. Specify which action code to use for each pair of primary and secondary event codes.

   The controls in the Action Assignment group box enable you to specify how the EDI agent translates between the event codes in a PeopleSoft business document and the action codes in the electronic commerce staging tables.

   Select one of the defined action codes in the EC Action Code field, and then select the primary and secondary event codes that map to this action code. If this transaction type uses only a primary event code (Pri Event field), you can leave the Sec Event (secondary event) box empty.

   Repeat this step to define each possible action code and combination of primary and secondary event codes for this transaction type.

4. Repeat steps 1 through 3 for each transaction type that partners with this profile can perform.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," EDI Codes and PeopleSoft Codes, page 5

Chapter 2, "Using PeopleSoft EDI Manager," Creating Transaction IDs, page 12

Setting Up Trading Partners

This section provides an overview of trading partners and discusses how to:

- Define PeopleSoft entity codes.
- Create internal partner IDs.
- Assign external trading partner IDs.
- Define business entities in external companies.

Understanding Trading Partners

In PeopleSoft EDI Manager, a trading partner is a business entity in an external company with which a business entity in your company does business.

When you set up trading partners in PeopleSoft EDI Manager, you need to define the internal structure of your company and, possibly, the companies with which you trade.

PeopleSoft Financials applications enable you to organize your company into multiple business units, each of which tracks its data somewhat independently. Each business unit likely does business with a different set of trading partners. When a trading partner submits an EDI transaction, the transaction must be addressed to a specific business unit or the EDI agent should know to which business unit to forward the transaction.

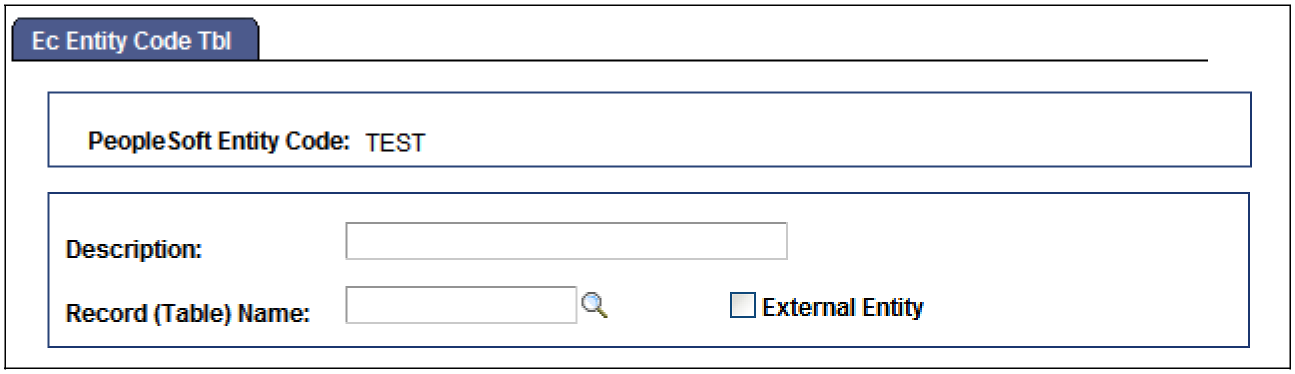
Conversely, the external companies that you do business with are not monolithic entities. They have different divisions, departments, or business units, and you might need to address EDI transactions to a specific entity inside that company.

Pages Used to Set Up Trading Partners

Page Name	*Definition Name*	*Navigation*	*Usage*
Ec Entity Code Tbl (Entity Code Table)	EC_ENTITY_CODE_TBL	PeopleTools, EDI Manager, Setup Trading Partners, Entity Codes	Define PeopleSoft entity codes.
Ec Int Partner Def (Internal Partner Definition)	EC_EXT_PARTNER_DEF	PeopleTools, EDI Manager, Setup Trading Partners, Internal Partners	Create internal partner IDs.
Ec Ext Partner Def (External Partner Definition)	EC_EXT_PARTNER_DEF	PeopleTools, EDI Manager, Setup Trading Partners, External Partners	Create external trading partner IDs.
Ec Bus Entity Def (Business Entity Definition)	EC_BUS_ENTITY_DEF	PeopleTools, EDI Manager, Setup Trading Partners, Business Entities	Define business entities in external companies.

Defining PeopleSoft Entity Codes

Access the Ec Entity Code Tbl (Entity Code Table) page.



PeopleSoft Entity Code: TEST

Description:

Record (Table) Name:

☐ External Entity

EC Entity Code Tbl page

You define entity codes for the types of business entities that serve as trading partners. Internally, the entities are usually business units. Use this page to specify the type of business entity to which the code applies and which table in the PeopleSoft database lists valid entities of this type.

To define a PeopleSoft entity code:

1. Select the record definition for the table that lists valid entities of this type.

   Select the prompt table for this type of entity from the Record (Table) Name field. For example, if you are defining an entity code for business units, specify the table that holds business unit IDs.

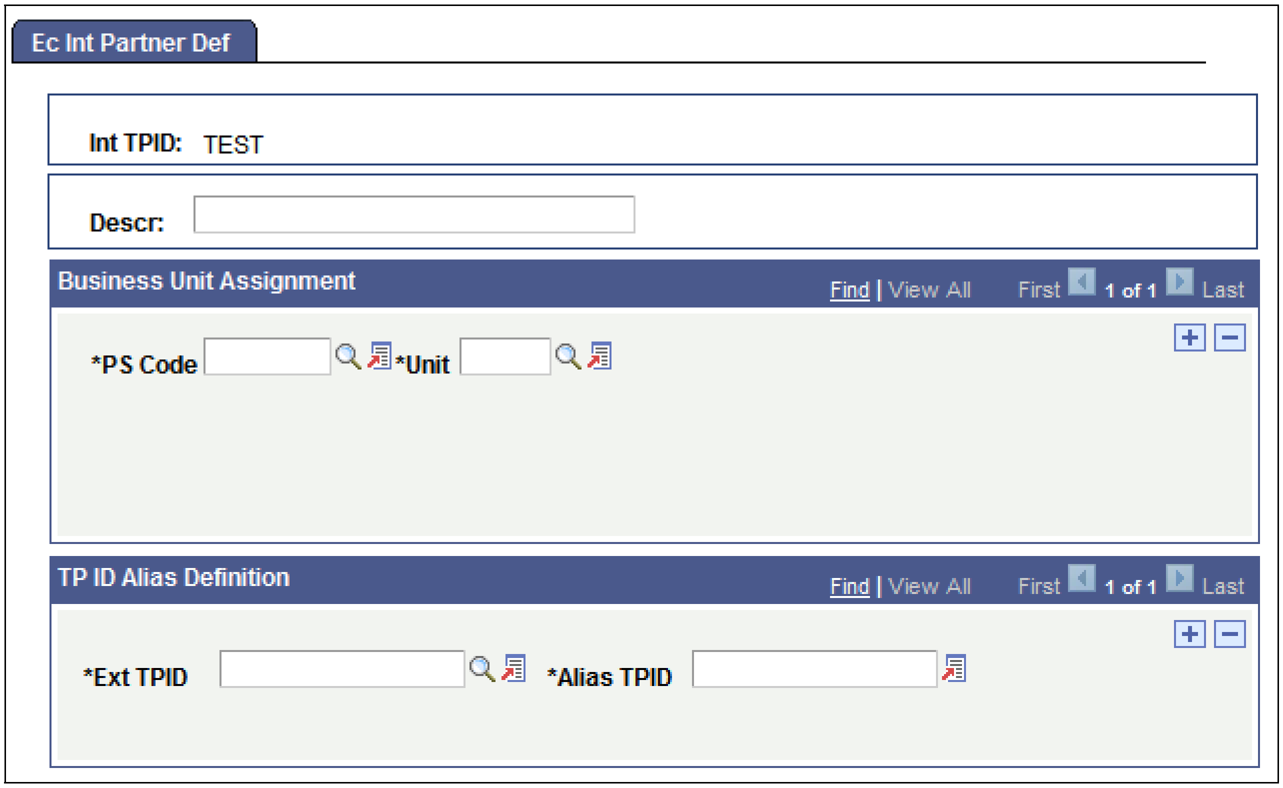
2. Specify whether this entity code applies to external trading partners.

   You need to identify entity codes both for your internal business entities and for the external companies that you trade with. For example, you will probably want to create entity codes based on the Vendor table and the Customer table.

   If the entity code that you are defining is for external companies, select the External Entity check box.

Creating Internal Partner IDs

Access the Ec Int Partner Def (Internal Partner Definition) page.


Ec Int Partner Def
Business Unit Assignment
Find | View All First 1 of 1 Last
TP ID Alias Definition
Find | View All First 1 of 1 Last

Int TPID: TEST

Descr:

*PS Code *Unit

*Ext TPID *Alias TPID

Ec Internal Partner Definition page

You assign trading partner IDs to the internal entities (business units) that external trading partners need to be able to submit transactions to. Because of the way each PeopleSoft Financials application tracks its own set of business units, a single trading partner ID can refer to multiple business unit IDs, all of which actually refer to the same physical business unit. Use this page to specify which internal business entities share this partner ID and which external trading partners do business with this partner.

To create an internal partner ID:

1. Specify to which business entity or entities this trading partner ID applies.

   In the PS Code field, select the entity code for the type of business entity, and then select the specific entity in the Unit field.

   Because the same business unit can be referred to by different business unit IDs in different PeopleSoft Financials applications, you can add multiple units to the same trading partner ID.

2. Specify which external trading partners work with this internal trading partner, and what name they use to refer to the internal partner.

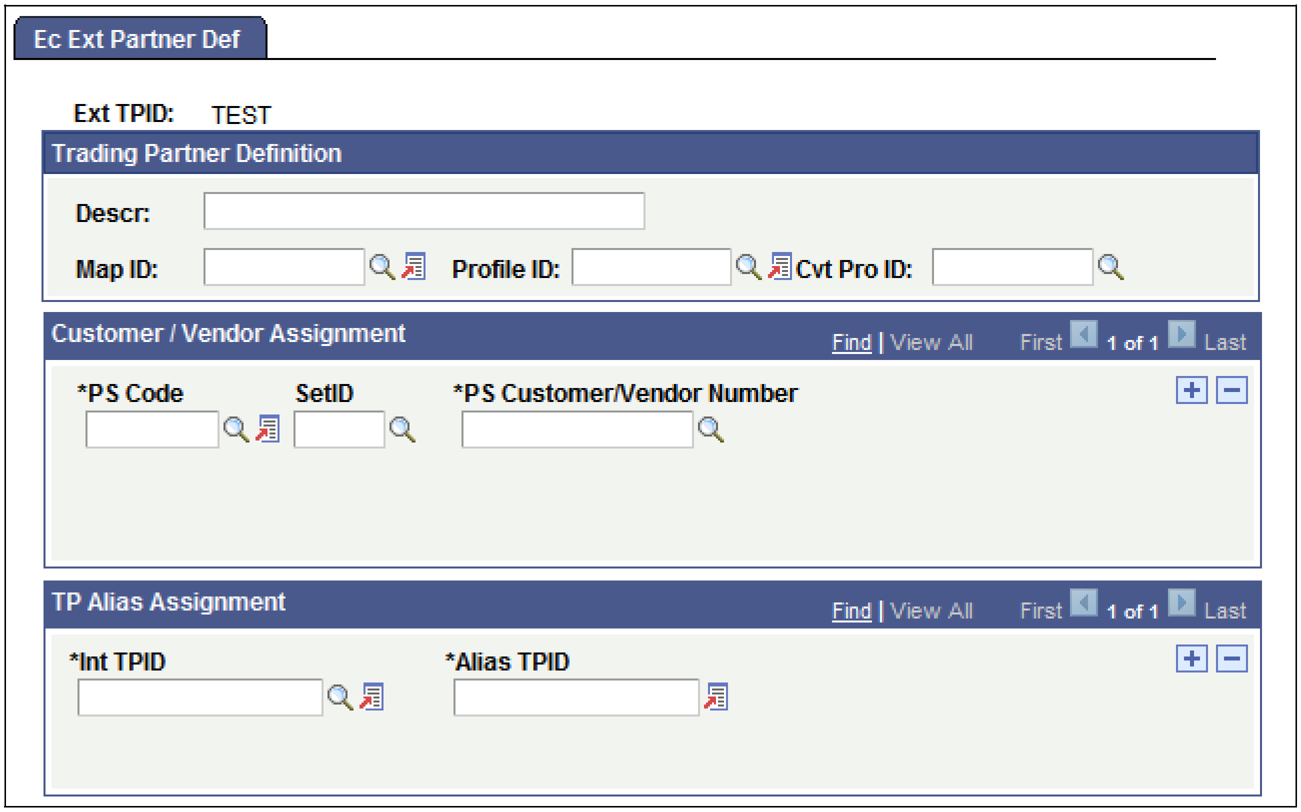
   In the Ext TPID field, select the name of an external trading partner that does business with the internal partner that you are defining. Then, in the Alias TPID field, enter the name by which the selected external partner refers to the internal partner, that is, the text that will appear in the Internal Trading Partner field in PeopleSoft business documents.

   Note. If you haven't defined your external trading partners yet, you'll need to return to this page after you add them.

   Repeat this procedure for each external trading partner.

Assigning External Trading Partner IDs

Access the Ec Ext Partner Def (External Partner Definition) page.


Ec Ext Partner Def

Ext TPID: TEST

Trading Partner Definition

Descr:

Map ID: Profile ID: Cvt Pro ID:

Customer / Vendor Assignment — Find | View All — First 1 of 1 Last

*PS Code — SetID — *PS Customer/Vendor Number

TP Alias Assignment — Find | View All — First 1 of 1 Last

*Int TPID — *Alias TPID

Ec External Partner Definition page

You'll need to find out from the trading partner or from the VAN what trading partner ID to use.

To add an external trading partner:

1. Select the map profile that includes the mappings that this trading partner uses.

   Select the appropriate map ID from the list of valid map profile IDs.

2. Select the partner profile that identifies the transactions that this partner can perform.

   Select the appropriate partner profile ID from the list of valid IDs.

3. (Optional.) Select the conversion data profile that specifies how to translate this partner's codes into PeopleSoft codes.

   Select the appropriate conversion data profile ID from the Cvt Pro ID field.

4. Specify the customer ID, vendor ID, or other ID for this trading partner.

   Note. If you exchange transactions with multiple business entities inside this trading partner, you'll need to define the partner's external business entities. If you define external business entities, each one needs to have its own unique customer ID, vendor ID, or other ID. The ID that you assign here should be the ID for the corporate office.

   In the PS Code field, select the entity code for the type of entity this partner is: customer, vendor, and so on. If your system uses table set processing, select the setID for the Customer or Vendor table in the SetID field. In the PS Customer/Vendor Number field, select the specific value from the Customer or Vendor table that corresponds to this trading partner.

   If different tables in the system refer to this trading partner by different names, you can list all the IDs as part of the trading partner definition.

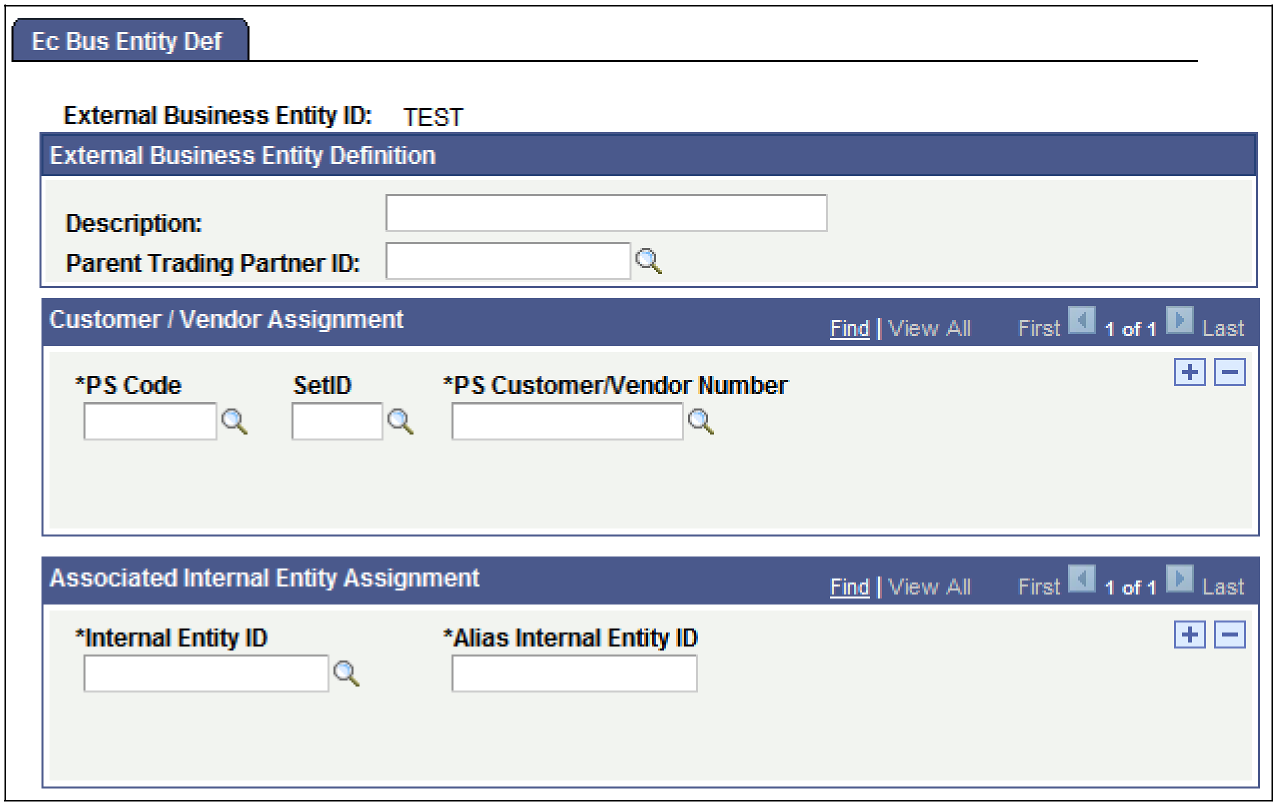
5. Identify the internal trading partners who do business with this external trading partner.

   In the Int TPID field, select the name of an internal trading partner that does business with the external partner you are defining. Then, in the Alias TPID field, enter the name by which the new external partner refers to the selected internal partner, that is, the text that will appear in the Internal Trading Partner field in the PeopleSoft business documents.

   Repeat this step for each internal trading partner.

Defining Business Entities in External Companies

Access the Ec Bus Entity Def (Business Entity Definition) page.


Ec Bus Entity Def

External Business Entity ID: TEST

External Business Entity Definition

Description:

Parent Trading Partner ID:

Customer / Vendor Assignment Find | View All First 1 of 1 Last

*PS Code SetID *PS Customer/Vendor Number

Associated Internal Entity Assignment Find | View All First 1 of 1 Last

*Internal Entity ID *Alias Internal Entity ID

Ec Business Entity Definition page

Perform this procedure only if you need to be able to address transactions to multiple business entities that are part of the same external trading partner. This is similar to creating an external partner definition, except that business entities are subordinate to external trading partners and acquire some of their properties. Use this page to define the groups or entities that you deal with inside an external trading partner.

To define business entities in an external trading partner company:

1. Specify which trading partner company this business entity is a part of.

   Select the appropriate trading partner ID from the Parent Trading Partner ID field.

   When the EDI agent processes a transaction that involves the business entity that you are defining, it will use the map ID, profile ID, and conversion data ID assigned to the parent trading partner.

2. Specify the customer ID or vendor ID for this business entity.

   If you want to establish a business entity as a valid trading partner, it must appear as a separate entry in your PeopleSoft database as a customer or a vendor. In the Customer/Vendor Assignment group box, you identify which customer or vendor is referred to by the external business entity ID that you are creating. The system needs this information to determine the customer ID or vendor ID to assign to incoming EDI transactions, and the external business entity ID to assign to outgoing EDI transactions.

   In the PS Code field, select the entity code for customers or vendors. If your system uses table set processing, select the setID for the Customer or Vendor table in the SetID field. In the PS Customer/Vendor Number field, select the specific value from the Customer or Vendor table that corresponds to this business entity.

   If different tables in the system refer to this business entity by different names, you can list all the IDs as part of the external business entity definition.

3. Identify the internal trading partners who do business with this external business entity, and then save the page.

   In the Internal Entity ID field, select the name of an internal trading partner that does business with the external entity you are defining. Then, in the Alias Internal Entity ID field, enter the name by which the new external entity refers to the selected internal partner, that is, the text that will appear in the Internal Trading Partner field in PeopleSoft business documents.

   Repeat this step for each internal trading partner.

Deleting PeopleSoft EDI Manager Objects

This section provides an overview of deleting PeopleSoft EDI Manager Objects and discusses how to delete PeopleSoft EDI Manager objects.

Understanding Deleting PeopleSoft EDI Manager Objects

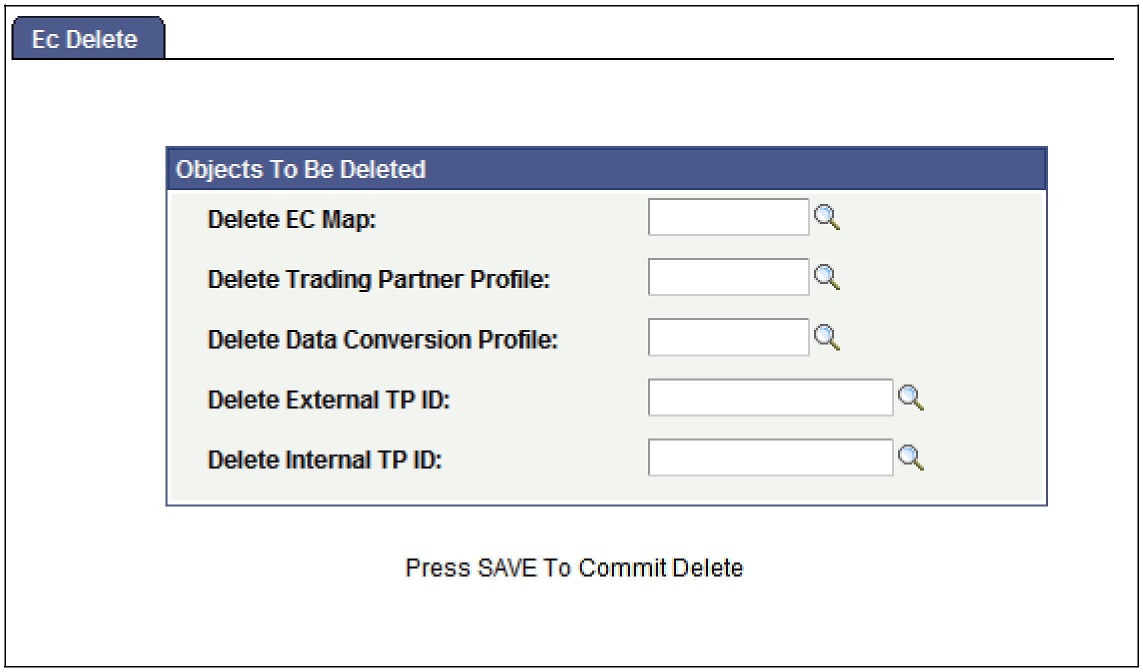
You might want to delete business entities if they are no longer valid. PeopleSoft enables you to delete maps, trading partner profiles, data conversion profiles, internal trading partners, external trading partners, and so on.

Page Used to Delete PeopleSoft EDI Manager Objects

Page Name	***Definition Name***	***Navigation***	***Usage***
Ec Delete	EC_DELETE	PeopleTools, EDI Manager, Setup Trading Partners, Remove EDI Manager Objects	Delete PeopleSoft EDI Manager objects.

Deleting EDI Manager Objects

Access the Ec Delete page.



Objects To Be Deleted

Delete EC Map:	
Delete Trading Partner Profile:	
Delete Data Conversion Profile:	
Delete External TP ID:	
Delete Internal TP ID:	

Press SAVE To Commit Delete

Ec Delete page

To delete EDI Manager objects, select the objects to be deleted in the appropriate field list, and then click Save to delete the selected objects.

Chapter 3

Mapping EDI Transactions

This chapter provides an overview of Electronic Data Interchange (EDI) transaction mapping and discusses how to:

- Define inbound maps.
- Define outbound maps.
- Create data mapping profiles.
- Use PeopleSoft EDI Manager for general data extraction.

Understanding EDI Transaction Mapping

This section discusses:

- EDI transaction mapping.
- New EDI transactions.
- PeopleSoft business document format.

EDI Transaction Mapping

PeopleSoft applications store transaction data in tables in the PeopleSoft database. For an application to process an EDI transaction, it needs the transaction data in its tables. So the first step in processing an EDI transaction is transferring the data from an incoming EDI Transaction Set file into the application tables. Similarly, the first step in generating an EDI transaction for delivery to a trading partner is getting the transaction data out of the application tables into an EDI Transaction Set file.

You specify how the EDI agent transfers data between EDI Transaction set files and application tables by creating electronic commerce maps. Two kinds of electronic commerce maps are available:

- Inbound maps, which transfer incoming transaction data into your PeopleSoft database.
- Outbound maps, which create outgoing EDI documents from transaction data in the database.

Map profiles specify the maps to which your trading partners have access.

New EDI Transactions

We deliver PeopleSoft applications with a number of EDI transactions already defined. You can use PeopleTools to add support for additional transactions. Here are the major steps in the process.

To process new EDI transactions:

1. Use PeopleTools to develop the application that processes the transaction.

   Create the database tables and the pages that users need to process the transaction.

2. Define a PeopleSoft Business Document format based on the EDI format for the transaction.

   At this step, think about how the data from the EDI document needs to map to the PeopleSoft application tables.

3. Write a translation program that converts EDI documents into PeopleSoft Business Document format.

   Usually, you will have an EDI translator program to do this step.

4. Create staging tables to serve as a temporary holding area between the PeopleSoft document and the application tables.

5. Create maps to copy data between the PeopleSoft business documents and the staging tables.

6. Write an application load procedure that transfers data from the staging tables to the application tables.

This chapter documents step 5 in this procedure.

PeopleSoft Business Document Format

A PeopleSoft business document is an ASCII file that contains the data for one or more EDI transactions. It contains transaction data and control information, which tells the system what type of transaction it is, who it came from, and how parts of the document relate to each other.

A PeopleSoft business document is divided into records, separated from each other with a carriage return/line feed (CR/LF) pair. Each record is divided into fixed-length fields. One of the fields in the record, usually the first one, is a record ID field that specifies the record type: a header, detail line, summary line, and so on. The remaining fields depend on the type of record.

The first record in a PeopleSoft business document is always a control record. The control record specifies the type of transaction contained in the following lines and the trading partners involved. Based on the control record, the EDI agent retrieves the mapping definition for the specified transaction and the data conversion options relevant to the trading partner.

The EDI agent reads the records in the document one at a time. For each record following the control record, the EDI agent reads the value in the record ID field and uses it to determine the PeopleSoft record definition to use to parse the rest of the record. The PeopleSoft record definition specifies the location, size, and data type of the remaining fields in the record. The EDI agent copies the data from the document into staging tables in the database, following the rules in the mapping definition.

When the EDI agent encounters a new control record, identified by the record ID *999* or *998,* it retrieves the mapping definition for the new transaction type and repeats the process. Within the flat file, the control records:

- Enable a third-party translator or trading partner, upon receiving the file, to determine what the data contains *(999* only).
- Enable PeopleSoft EDI Manager, upon receiving the file, to determine what the data contains *(999* and *998).*

Most transactions include a variety of record types: a header, detail lines, schedules, summary lines, and so on. The layout of the records within a PeopleSoft business document follows a logical pattern. All detail lines linked to a particular parent or header line must follow the parent record.

Note. The EDI agent processes transactions at the "unit of work" level, corresponding to the ST/SE level in X.12 format. Each transaction should be a unit of work that you want the EDI agent to commit or roll back as a unit, such as a single purchase order or invoice.

Let's look at an example:

```
001 Header Line 1
002 Detail Line 1A
002 Detail Line 1B
002 Detail Line 1C
001 Header Line 2
002 Detail Line 2A
002 Detail Line 2B
```

This file has two header lines and five detail lines. The EDI agent determines which detail lines go with which header lines based entirely on their order.

When the EDI agent processes this PeopleSoft business document, it works through the records in order. First, it processes header line 1 using the work record that's appropriate for record ID 001. Next, it processes detail line 1A using the work record for record ID 002.

If you ask the EDI agent to calculate a value, such as the sum of all the detail lines, it determines which detail lines to group together based on their position. It calculates the value for all detail lines *since the most recently encountered header line.*

999 Record Format

This reserved record identifier acts as a control record to switch map definitions within a data file. The layout looks like this :

Field	***Description***	***Value***
1–3	Record Identifier.	*'999'*
4–18	Transaction ID.	*Char*
19–22	External entity code.	*Char / Values :* *CUST* or *VNDR*
23–38	External trading partner ID.	*Char*

Field	***Description***	***Value***
39–42	Internal alias entity code.	*Char / Values :* *AP,AR,OM, ...*
43–58	Internal trading partner alias ID.	*Char*

998 Record Format

This format allows the specification of a PeopleSoft map ID directly in the data contents. This type of map definition can be used when trading partner conversions are not needed or not available. All other data conversion functionality can be used.

Field	***Description***	***Value***
1–3	Record identifier.	*'998'*
4–13	PeopleSoft EC map ID.	*Char*
14–23	PeopleSoft trading partner profile ID.	*Char*
24–33	PeopleSoft data conversion profile ID.	*Char*

Defining Inbound Maps

This section describes how to:

1. Create staging tables and work records.
2. Create inbound maps.
3. Define the inbound PeopleSoft business document format.
4. Specify staging tables.

You create one map for each EDI transaction.

Note. Electronic commerce maps help transfer data between PeopleSoft business documents and staging tables in a PeopleSoft database. They don't apply to the application load or application extract processes, which copy data between the staging tables and the transactional tables.

Pages Used to Define Inbound Maps

Page Name	***Definition Name***	***Navigation***	***Usage***
Inbound Maps - Description page	EC_MAP_00	PeopleTools, EDI Manager, Map EDI Transactions, Inbound Maps Select the Description tab.	Create inbound transaction maps.
Inbound Maps - Business Document Layout	EC_INBOUND_MAP_01	PeopleTools, EDI Manager, Map EDI Transactions, Inbound Maps Select the Business Document Layout tab.	Define business document layouts.
Inbound Maps - Target Records	EC_INBOUND_MAP_02	PeopleTools, EDI Manager, Map EDI Transactions, Inbound Maps Select the Target Records tab.	Specify staging tables.

Creating Staging Tables and Work Records

To create staging tables and work records:

1. In Application Designer, create the staging tables for the incoming data.

   Save the record definitions with the extension _EC to identify them as staging tables. Use SQL Create to create tables using the record definitions.

   The tables should be structured as an intermediate step between the PeopleSoft Business Document format that you want to translate and the structure of the transaction tables.

   The record definition for a staging table has three kinds of fields:

   - Required system fields, which must be key fields for the tables.

Field Name	***Description***
ECTRANSID	Transaction ID.
ECQUEUEINSTANCE	Instance ID.
ECTRANSINOUTSW	*I* or *O*, specifying an incoming or outgoing transaction.

   - Fields corresponding to the data fields in the PeopleSoft business documents.
   - (Optional) Extra fields for calculated data needed in the application tables, such as summary values, date stamps, or action codes.

2. In PeopleSoft Application Designer, create work records that mimic the structure of the records in the PeopleSoft business document that you want to translate.

   You need to create a work record for each physical record in the PeopleSoft business document file.

   One field in the work record needs to be designated as the record ID for the document record that it mimics. For most records, it's the first field: the record ID in the PeopleSoft business document is usually the first data on the line.

   The EDI agent uses the record ID in the work record to match the appropriate work record with each record in the PeopleSoft business document. You can add the appropriate record ID as a default value for the field; PeopleSoft EDI Manager will automatically supply the correct record ID when you describe the business document format on the Business Document Layout page.

   The rest of the fields in the work record definition should match the corresponding fields in the PeopleSoft Business Document record. They should be in the same order and should have the same length. Add default values for any fields that have them.

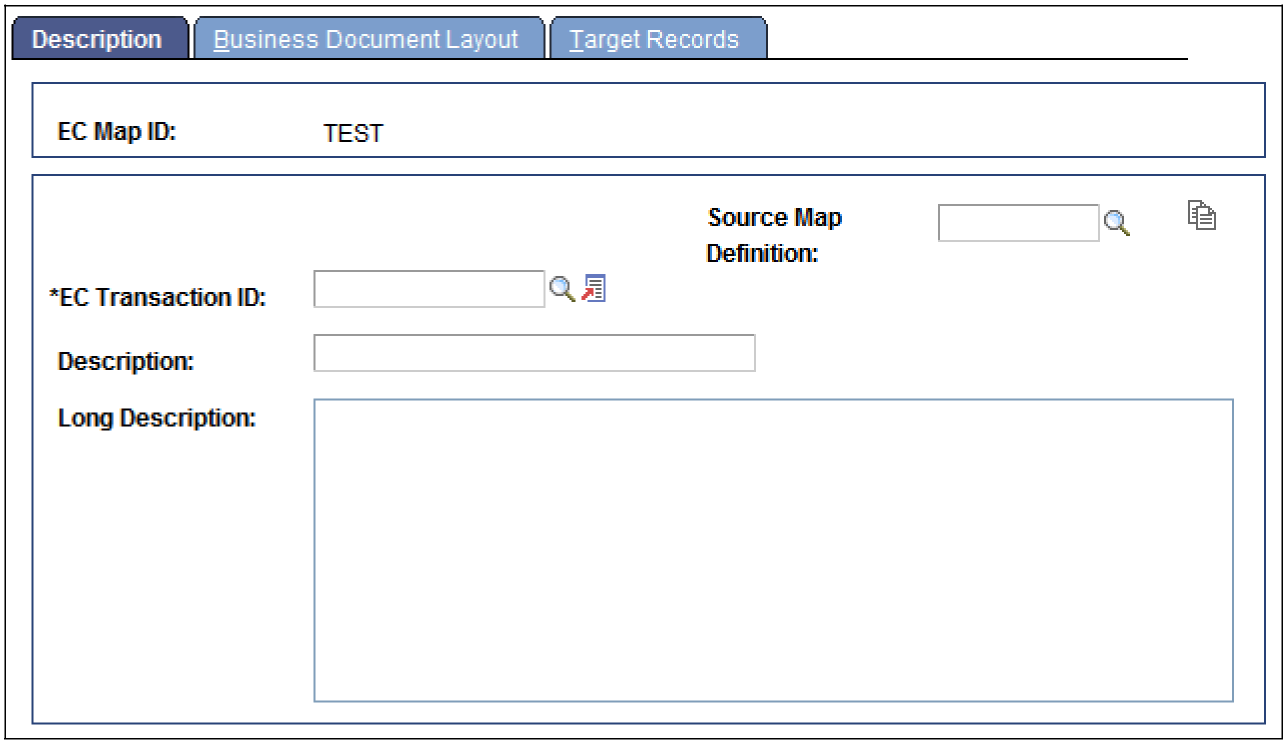
   Save the work record with the extension _WD to identify it as a work document.

See Also

Chapter 3, "Mapping EDI Transactions," PeopleSoft Business Document Format, page 26

Creating Inbound Maps

Access the Inbound Maps - Description page.



EC Map ID: TEST

Source Map Definition:

*EC Transaction ID:

Description:

Long Description:

Inbound Maps - Description page

To create an inbound map:

1. In the EC Transaction ID field, select the transaction to which this map applies.
2. Enter a description of the map.

   You can enter both a short description (in the Description text box) and a long description (in the larger box). These descriptions are for your information only; the system does not use them.
3. (Optional.) Copy the definition of a similar map.

   You can copy the details of an existing map definition into this component and change just the parts that are different.

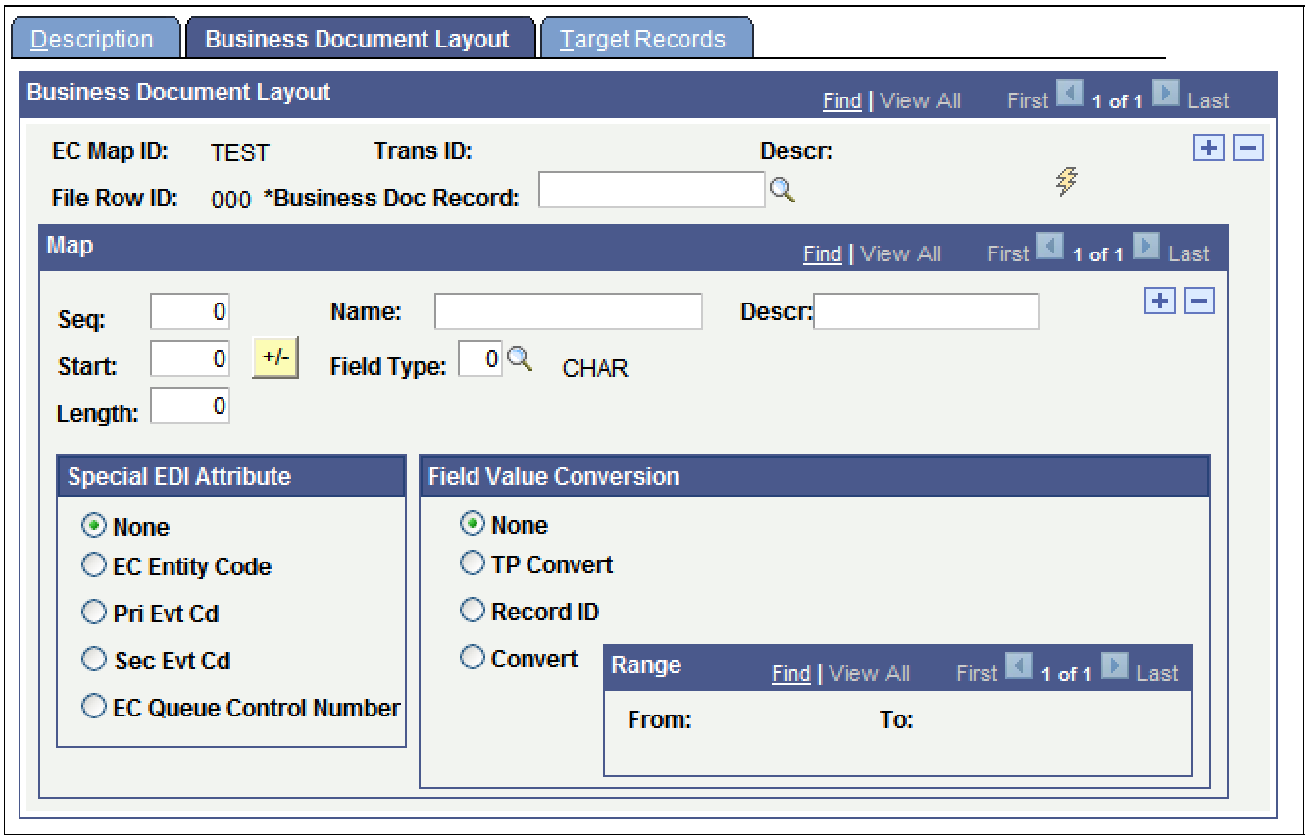
   Select the existing map in the Source Map Definition field, and then click the Ec Copy Map DefnSwitch button. The system copies the information from the selected map into this page.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," page 3

Defining the Inbound PeopleSoft Business Document Format

Access the Inbound Maps - Business Document Layout page.



EC Map ID: TEST
Trans ID:
Descr:
File Row ID: 000
*Business Doc Record:

Seq: 0
Name:
Descr:
Start: 0
Field Type: 0 CHAR
Length: 0

Special EDI Attribute: [✓ None] EC Entity Code / Pri Evt Cd / Sec Evt Cd / EC Queue Control Number

Field Value Conversion: [✓ None] TP Convert / Record ID / Convert

From:
To:

Inbound Maps - Business Document Layout page

You have already created work record definitions that mimic the structure of the PeopleSoft business documents. On this page, you specify which work record definition to use for each record in the PeopleSoft business documents and what type of data appears in each field.

To define the inbound format:

1. Select a work record definition from the Business Doc Record field, and then click the EC Get File Fields button.

   PeopleSoft EDI Manager copies data from the selected record definition into the Map group box, with one row for each field in the record.

2. Specify the record ID for this work record definition.

   If the first field in the work record definition is not the one for record IDs, browse the rows in the Map group box to find the record ID field. The EDI agent uses this field to match the work record definition to the appropriate records in PeopleSoft business documents.

   a. In the Map group box for the appropriate record ID, select the Record ID option in the Field Value Conversion group box. The EC File Row ID Value text box appears.

   b. In the text box, enter the record ID of the records whose structure this work record definition mimics. When the EDI agent finds a record in a PeopleSoft business document with this record ID, it uses this work record definition to parse it.

3. Select the next field in the work record definition.

4. Check the location of a field in the record.

   If you defined the work record definition properly, the correct field information appears in the text boxes, copied from the work record. You can verify the existing information rather than entering it. If the format of the PeopleSoft business document is slightly different than the work record definition, you can modify the formatting on this page to match the document without affecting the work record definition.

   The Seq (sequence) text box gives the number of the field in the record. The first field in the record is field 1, the second is field 2, and so on. The Name field is the field name from the work record, and the Field Type field indicates the data type of the field's data.

   The Start and Length boxes specify the location and size of the field in the PeopleSoft business document. The start position is the number of characters from the beginning of the record to where the first character of this field's data appears. The length is the number of characters reserved for this field; it's also the number of characters that the system will transfer to the staging tables.

   Note.

   Length is defined in characters, not in bytes.

   For numeric data, the Dec (decimal positions) field indicates the number of characters to the right of the decimal point. If an incoming PeopleSoft business document uses implicit decimal format, that is, if it does not include decimal points in this field (in any row), the EDI agent inserts them (in all rows) at the position specified in this box. If the data already includes decimal points, the EDI agent does not change them or insert any other decimal points.

   For date information, you also need to enter a date format and delimiter. You use these text boxes to specify the format for dates in this field in the incoming file. Use standard date formatting conventions, as shown in this table:

Date Value	***Date Format***	***Delimiter***
19961226	YYYYMMDD	N (for *none*)
1996/12/26	YYYYMMDD	/
DEC–96–26	MMMYYDD	–
26–DEC–1996	DDMMMYYYY	–
26.12.1996	DDMMYYYY	–

5. Select an option in the Special EDI Attribute group box to specify whether the data in the current field is transaction data or an EDI control code.

   Options are:

None	Normal transaction data.
EC Entity Code	An entity code specifies the type of business entity that the trading partner ID for the current transaction refers to. The entity code tells the EDI agent that the table to use to find the list of valid IDs.
EC Queue Control Number	A unique identifying number that you want to store in the EDI agent queue and refer to when reviewing EDI audit history. For example, in a purchase order, you might identify the PO number as the EC queue control number.
Pri Evt Cd (primary event code) and ***Sec Evt Cd*** (secondary event code)	The primary event code, sometimes called the purpose code. The EDI agent uses event codes to determine the appropriate action code. If you identify a primary event code, you must also identify a secondary event code, sometimes called the transaction code, in the same record.

6. Specify how the EDI agent needs to convert the data as it copies data from the field into the staging tables.

   Field value conversion options are:

None	The EDI agent copies the data exactly as it appears in the PeopleSoft business document field.
TP Convert	The EDI agent converts data from this field according to the rules defined in a conversion type definition. When you select this option, a drop-down list appears to select a conversion type definition.
Record ID	Use this option for the first field in the work record definition only. (See step 3.)
Convert	The EDI agent converts the data as specified in the From and To boxes. When the field in the PeopleSoft business document contains the value in the From box, the EDI agent converts it into the value in the To box. Enter a row for each possible value in the field.

Note. You can perform data conversion at several points during processing. Use the Convert option for conversions that apply to this map only and are valid for all trading partners.

7. Repeat steps 4 through 6 for each field in the work record definition.

8. Repeat steps 1 through 7 for each record in the PeopleSoft business document.

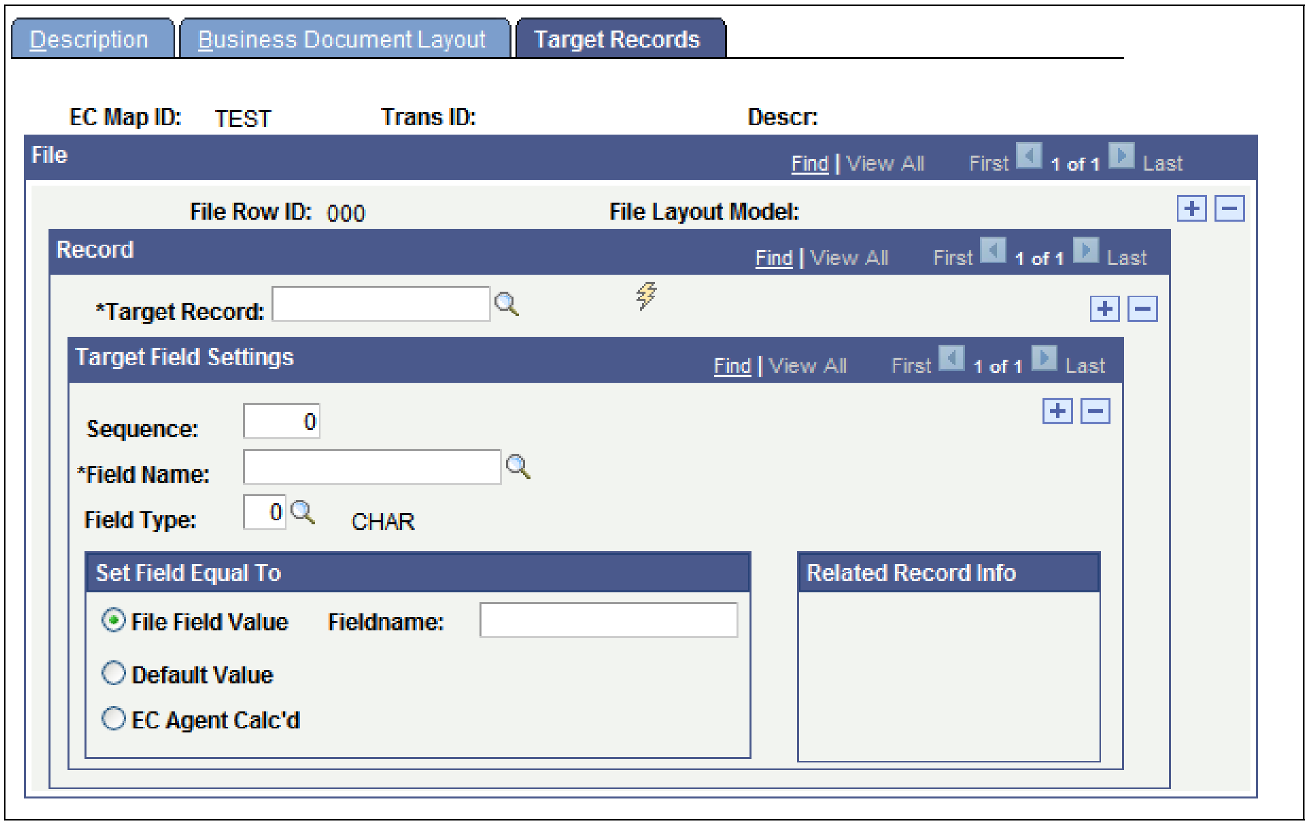
   To associate another work record definition with this map, put the cursor in the Business Doc Record field and press the F7 key.

See Also

Chapter 2, "Using PeopleSoft EDI Manager," page 3

Specifying Staging Tables

Access the Inbound Maps - Target Records page.



EC Map ID: TEST Trans ID: Descr:

File — Find | View All First 1 of 1 Last

File Row ID: 000 File Layout Model:

Record — Find | View All First 1 of 1 Last

*Target Record:

Target Field Settings — Find | View All First 1 of 1 Last

Sequence: 0

*Field Name:

Field Type: 0 CHAR

Set Field Equal To

(●) File Field Value Fieldname:

( ) Default Value

( ) EC Agent Calc'd

Related Record Info

Inbound Maps - Target Records page

To specify staging tables:

1. Select a work record definition.

   Browse the rows in the File group box to select a work record definition that you added to this mapping on the previous page. The record definition name appears in the File Layout Model field.

2. In the Target Record field, select the staging table to copy data into, and then click the EC Get Recfields SW button.

   Field information is copied from the record definition into the page.

3. Select a field in the staging table record definition.

   Browse the rows in the Target Field Settings group box to select the field.

   The Sequence field gives the position of the field in the record. The field name is from the record definition, and the field type indicates the data type of the field's data. All of this information comes from the record definition that you selected.

4. In the Set Field Equal To group box, select the value that you want the EDI agent to copy into the selected field.

   To copy the value from a field in the PeopleSoft business document, select the File Field Value option. Enter the name of the field whose value you want to copy in. Use the field name from the work record definition that appears as the file layout model at the top of the page.

   To enter a default value into the table, select the Default Value option and enter the value in the text box that appears.

   Note. You can specify both a file field to copy data from and a default value. The EDI agent enters the default value into the staging table only when the specified file field does not have a value.

   To enter a value that the EDI agent calculates as it copies data into the staging table, select the EC Agent Calc'd (EC agent calculated) option, and then select a calculation option from the drop-down list box that appears. Options include:

Calculation Option	***Value Written in the Field***
Action Code Conversion	An action code. The EDI agent gets the values from the fields in this record labeled Primary Event Code and Secondary Event Code, looks up the combination in the partner profile for the trading partner, and inserts the appropriate action code.
Apportion Parent Value	A percentage of the value from the parent line. The EDI agent takes the numeric value from the preceding parent line, divides it by the number of child lines, and puts the resulting value in this field on each child line. You have to complete out the Related Record Info group box to identify the parent line.
Average Summary Value	The average value from a particular field. The EDI agent adds up the values in this field from all occurrences of this record reporting to the same parent line, divides by the number of records, and enters the result in this field. You have to complete the Related Record Info group box to identify the field whose average you want.
Business Document Level External TPID (business document level external trading partner ID)	The trading partner ID of the trading partner that generated the EDI transaction. The EDI agent gets the trading partner ID from the first record in the PeopleSoft business document.

Calculation Option	***Value Written in the Field***
Business Document Level Internal TPID (business document level internal trading partner ID)	The trading partner ID of the internal group to which the EDI transaction is addressed. The EDI agent gets the trading partner ID from the first record in the PeopleSoft business document.
Current Date	The date on which the EDI agent processes the PeopleSoft business document.
Current Date and Time	The date and time when the EDI agent processes the PeopleSoft business document.
Current Time	The time when the EDI agent processes the PeopleSoft business document.
EC Queue Instance	The system-generated queue instance ID given to this PeopleSoft business document. **Note.** To create a valid mapping, you must map this value to the ECQUEUEINSTANCE field.
EC Transaction ID	The transaction ID from the first record in the PeopleSoft business document. **Note.** To create a valid mapping, you must map this value to the EDTRANSID field.
File Name/File ID	The name of the PeopleSoft business document file.
Incremented Key Sequence Number	A system-generated sequence number. The EDI agent increments the number for each child row of a particular parent. It starts at one again when it reaches a new occurrence of the parent line. You have to complete the Related Record Info group box to identify the parent line.
Inherit Parent Value	A value copied directly from the parent line. You have to complete the Related Record Info group box to identify the parent line.

Calculation Option	**Value Written in the Field**
Maximum Summary Value	The maximum value of a field. The EDI agent checks all the occurrences of a specified record and field and copies the maximum value into this field. You have to complete the Related Record Info group box to identify the field whose maximum you want.
Minimum Summary Value	The minimum value of a field. The EDI agent checks all the occurrences of a specified record and field, and copies the minimum value into this field. You have to complete the Related Record Info group box to identify the field whose minimum you want.
Operator ID	The ID of the user who started the EDI agent.
Process Instance	The system-generated instance ID from PeopleSoft Process Scheduler.
Run Control ID	The run control ID of the run control used to start the EDI agent.
Total/Accumulate Summary Value	The total of the values in a field. The EDI agent adds the values from a specified field in all occurrences of a record reporting to the same parent line and enters the result in this field. You have to complete the Related Record Info (related record information) group box to identify the field whose total you want.
Trading Partner Conversion	The customer, vendor, or business unit that is associated with a trading partner ID.

5. Enter the related record information (for some calculated options only).

   For some of the calculation options that you just selected, you identify the row, record, and field to use in the calculation.

   For example, if you selected the Total/Accumulate Summary Value option, the EDI agent adds up the values from the specified row, record, and field.

   Note. If you selected the Increment Key Sequence Number option, only the Row field appears. You don't need to select a record or field.

6. Repeat steps 3 through 5 for each field in the staging table.

   To create a valid mapping, you must enter the appropriate values in the three required system fields:

   - ECTRANSID: Use the EC Transaction ID option to have the EDI agent put the transaction ID in this field.
   - ECQUEUEINSTANCE: Use the EC Queue Instance ID option to have the EDI agent put the instance ID in this field
   - ECTRANSINOUTSW: Enter *I* to specify an incoming transaction.

7. Repeat steps 2 through 6 for each staging table that you want to copy data into.
8. Repeat steps 1 through 7 for each work record definition.

Defining Outbound Maps

This section discusses how to:

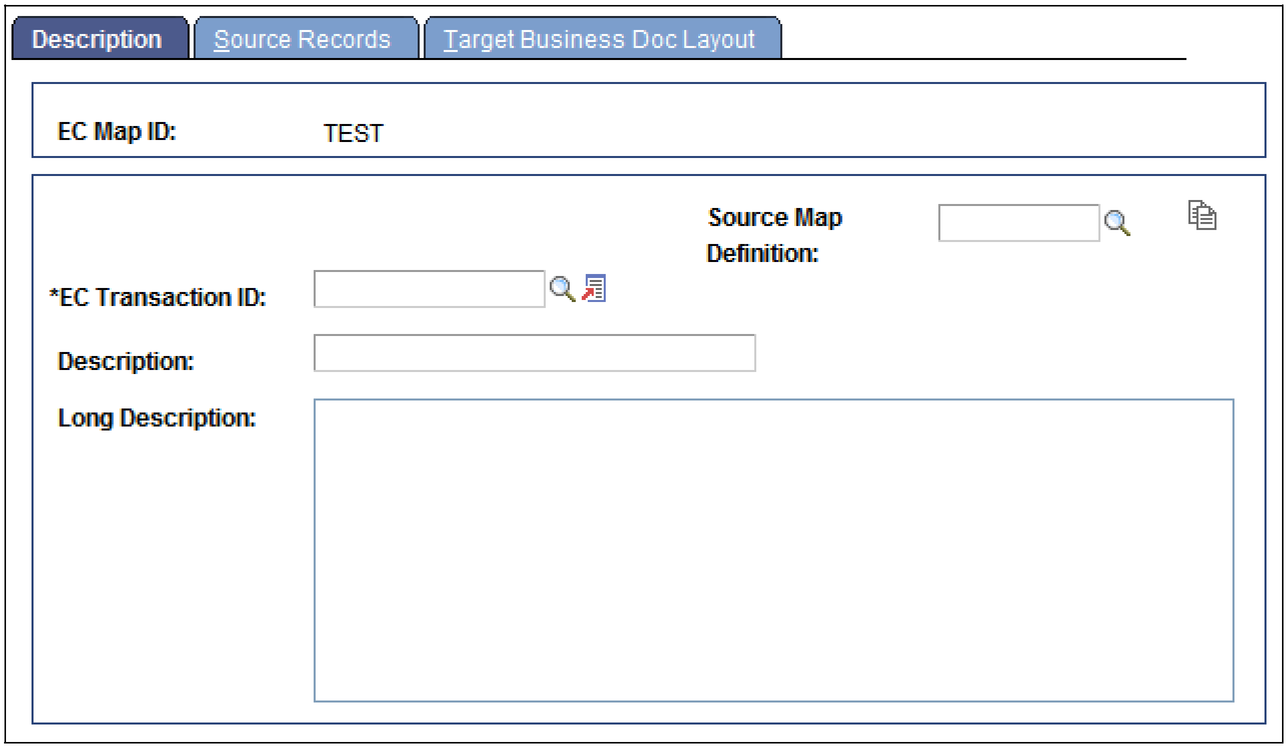
1. Create outbound maps.
2. Specify source records.
3. Define the outbound PeopleSoft Business Document format.

Pages Used to Define Outbound Maps

Page Name	***Definition Name***	***Navigation***	***Usage***
Outbound Maps - Description	EC_OUTBOUND_MAP_01	PeopleTools, EDI Manager, Map EDI Transactions, Outbound Maps Select the Description tab.	Create an outbound transaction map.
Outbound Maps - Source Records	EC_OUTBOUND_MAP_01	PeopleTools, EDI Manager, Map EDI Transactions, Outbound Maps Select the Source Records tab.	Specify source records.
Outbound Maps - Target Business Doc Layout (Target Business Document Layout)	EC_OUTBOUND_MAP_02	Select the Target Business Layout Doc tab.	Define the outbound PeopleSoft Business Document format.

Creating Outbound Maps

Access the Outbound Maps - Description page.


Description | Source Records | Target Business Doc Layout

EC Map ID: TEST

Source Map Definition:

*EC Transaction ID:

Description:

Long Description:

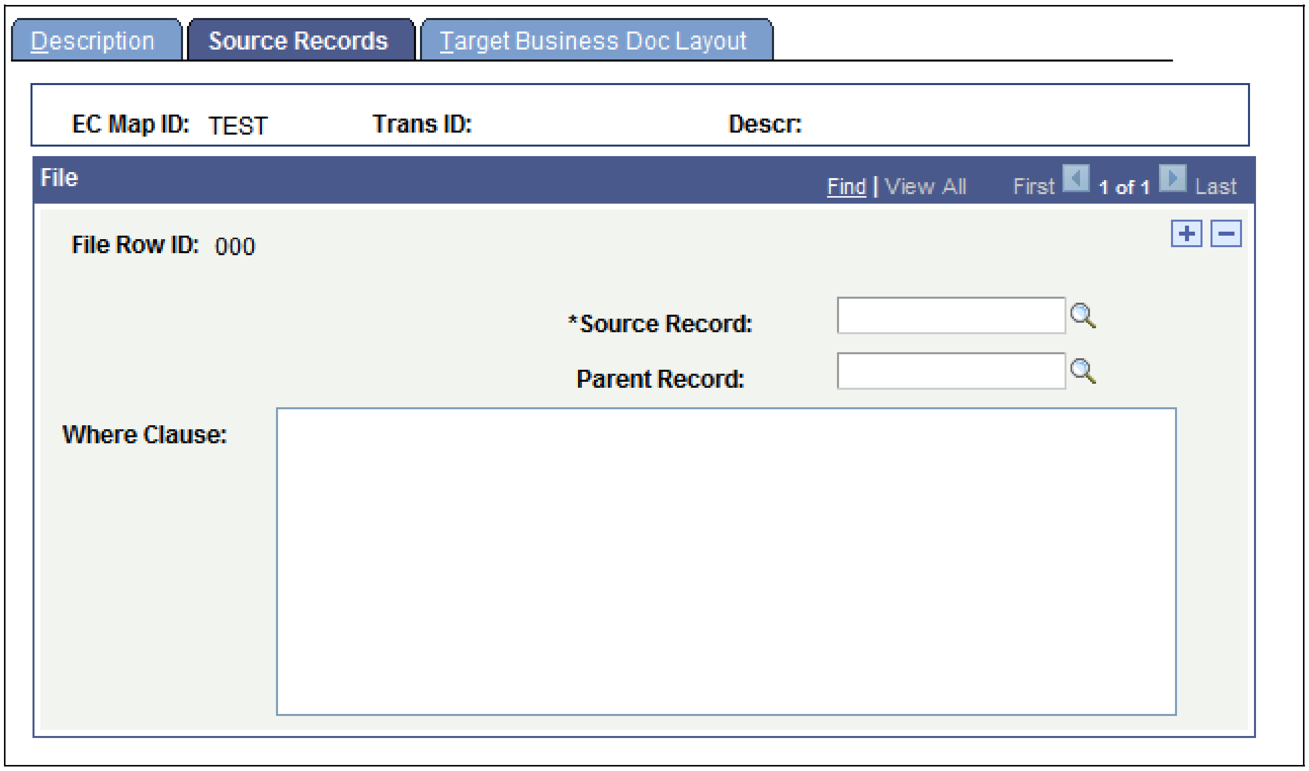
Outbound Maps - Description page

The initial procedure for creating an outbound map is the same as that for creating an inbound map.

See Chapter 3, "Mapping EDI Transactions," Creating Inbound Maps, page 31.

Specifying Source Records

Access the Outbound Maps - Source Records page.



EC Map ID: TEST Trans ID: Descr:

File Row ID: 000

*Source Record:

Parent Record:

Where Clause:

Outbound Maps - Source Records page

To specify source records:

1. In the Source Record field, select the record definition that retrieves the data that you want from the staging tables.

   You might select the record definition used to define the table or a view that retrieves just the data that you want.

   Add lines to the PeopleSoft business document in a logical unit of work order: header information before detail line information, parent lines before child lines, and so on.

   Note. You don't need to create a control record for the PeopleSoft business document (the 999 or 998 record). The EDI agent creates one automatically from information in the queue that stores pending outbound transactions.

2. Select a parent record (child lines only).

   If the record definition that you selected in the previous step retrieves data for a record that is related to a preceding parent record, such as the detail line under a header, select the record definition for the parent record in the Parent Record field.

   The parent record is typically the record definition that you used for the previous row in the outbound mapping.

3. Enter the SQL Where clause to specify which rows to retrieve.

   When you're creating a header line, the Where clause needs to select the header information for a single transaction. You identify a single transaction using the transaction ID and queue instance ID. The Where clause should look like this:

   ```
   WHERE ECTRANSID = $ECTRANSID$ AND ECQUEUEINSTANCE = $ECQUEUEINSTANCE$
   ```

   The two values inside dollar signs are system variables that the EDI agent replaces with the transaction ID and queue instance ID, respectively, of the first transaction in the pending outbound queue. This Where clause retrieves only the data for one transaction.

   If you're extracting the data for child lines, the Where clause must include a link to the parent record to tell the EDI agent how to join the tables. The Where clause needs to include a statement of this form:

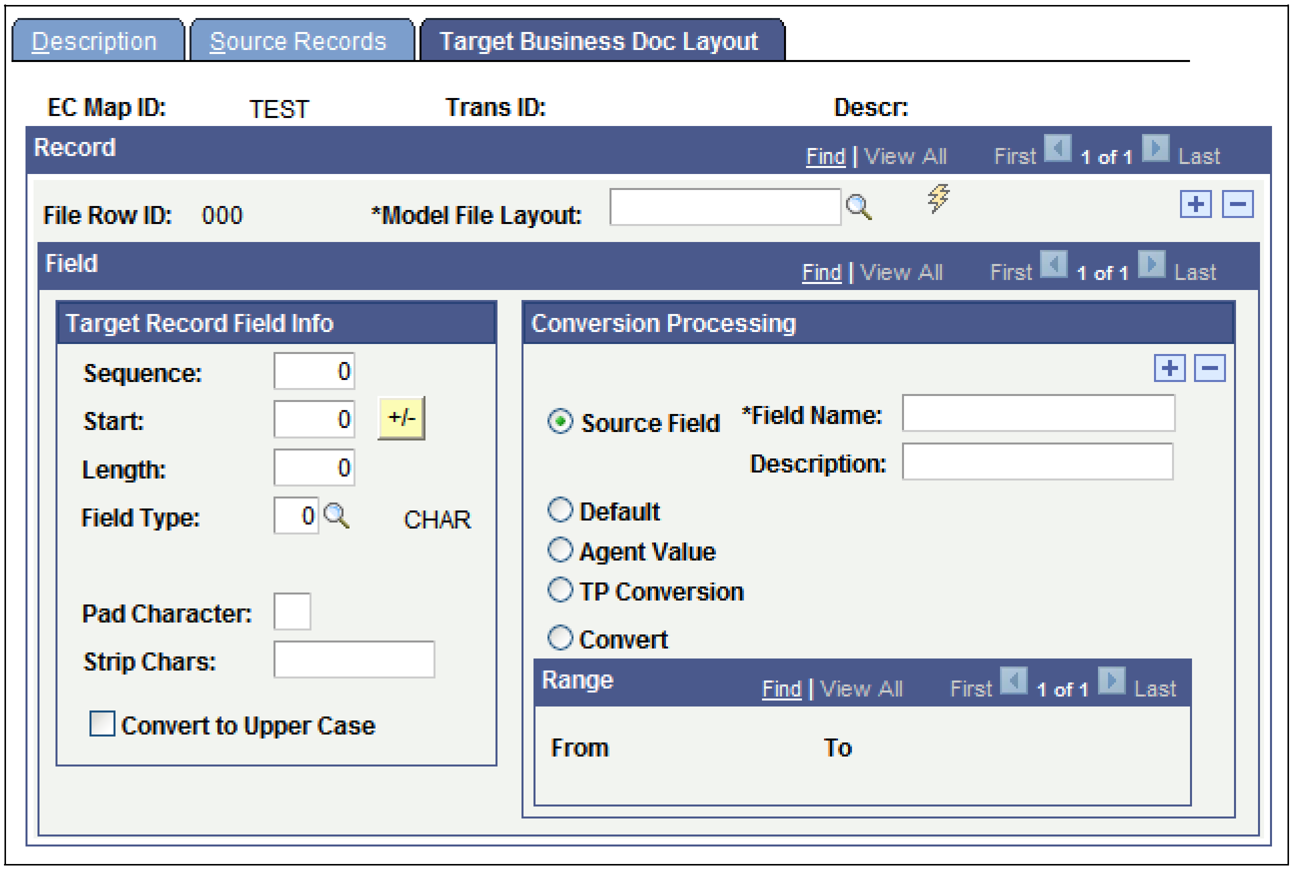
   ```
   WHERE fieldname = $record.fieldname$
   ```

   The first *fieldname* is a field from the source record definition; *record.fieldname* is a field in the parent record definition. You must include the dollar signs around *record.fieldname.*

4. Repeat steps 1 to 3 to specify the record definition for each record to go into the PeopleSoft business document.

Defining the Outbound PeopleSoft Business Document Format

Access the Outbound Maps - Target Business Doc Layout page.



EC Map ID: TEST Trans ID: Descr:

Record

File Row ID: 000 *Model File Layout:

Field

Target Record Field Info

Sequence: 0

Start: 0

Length: 0

Field Type: 0 CHAR

Pad Character:

Strip Chars:

☐ Convert to Upper Case

Conversion Processing

[✓ Source Field] *Field Name:

Description:

Default

Agent Value

TP Conversion

Convert

Range

From To

Outbound Maps - Target Business Doc Layout page

For each of the source record definitions that you added on the previous page, you identify an associated work record definition that specifies how the EDI agent writes the staging table data into PeopleSoft business documents.

To define the outbound format:

1. Select the work record definition that specifies the format of the output record, and then click the Get File Fields SW button.

   The File Row ID field identifies a source record definition from the previous page. In the Model File Layout field, select the work record definition that the EDI agent will use to write data from the source record into PeopleSoft business documents.

   When you click the Get File Fields SW button, the system copies field information from the work record definition into the page. The *Target Record Field Info* group box displays information about where in the record it will write each field's data.

   Depending on the data type of the record field, the Target Record Field Info group box offers additional formatting options so that you can format the data in the outgoing PeopleSoft business document.

   - For any field, enter in the Strip Chars field any characters that you don't want the outbound EDI agent to include when it copies data into the outbound transaction file. For example, you may want to strip the hyphens from phone numbers and Social Security numbers, or the decimal from a currency amount. When the outbound EDI agent writes data from a field into an outbound transaction file, it first removes all characters in the Strip Chars field.

     Note. The Strip Chars field contains a list of the individual characters to remove, not a string to remove. For example, if the field data is *455-67-8898* and the Strip Chars field has - (a hyphen) in it, the EDI Agent writes *455678898* to the file. If the Strip Chars field has the string -*5* in it, the EDI Agent writes the field data to the file as *4678898.*

   - For character fields, select the Convert to Upper Case check box to tell the outbound EDI agent to write the contents of the field in all uppercase letters.

   - For character fields or numeric fields, use the Pad Character field to specify a character for the outbound EDI agent to use to pad data so that it is the full length of the field. For character fields, the outbound EDI agent adds the specified character to the left of the existing field information. For numeric fields, the most typical use is to pad a number with initial zeros.

   - For date fields, you can specify in the Date Fmt (date format) and Delimiter fields how to format dates in this field in the business document. Use standard date formatting conventions, as illustrated in the procedure for creating inbound maps.

     See Chapter 3, "Mapping EDI Transactions," Defining the Inbound PeopleSoft Business Document Format, page 32.

2. Specify the data that you want the EDI agent to enter in the field.

   In the Conversion Processing group box, select the value that you want the EDI agent to enter into the selected field.

   - To copy the value from a field in the source record, select the Source Field option. Enter the name of the field whose value you want to copy in.

     Note. You need to specify a source field for all fields in the work record definition except those for which you provide a default value. You must select a source field even if you select another option as well.

   - To enter a default value into the document, select the Default option and enter the value in the field that appears.

     You can specify both a file field to copy data from and a default value. If you do, the EDI agent enters the default value into the document only when the specified field does not have a value.

   - If you select TP Conversion,, the EDI agent converts data from the specified source field according to the rules defined in a conversion type definition. When you select this option, a drop-down list box appears to select a conversion type definition.

   - If you select Convert, the EDI agent converts the data in the source field as specified in the From and To fields that appear. When the field in the PeopleSoft business document contains the value in the From field, the EDI agent converts it into the value in the To box. Enter a row for each possible value in the field.

- To enter a value that the EDI agent calculates as it copies data into the staging table, select the Agent Value option, and then select a calculation option from the drop-down list box that appears. The following table describes the available calculation options.

Calculation Option	***Value Written in the Field***
Action Code/Primary Event Code	An action code. The EDI agent gets the values from the fields in this record labeled Primary Event Code and Secondary Event Code, looks up the combination in the partner profile for the trading partner, and inserts the appropriate action code.
Current Date	The date on which the EDI agent creates the PeopleSoft business document.
Current Datetime	The date and time when the EDI agent creates the PeopleSoft business document.
Current Time	The time when the EDI agent creates the PeopleSoft business document.
EC Entity Code Flag	Flags this field as an entity code field to use in determining trading partner conversion logic. The values specify whether the entity is a customer, vendor, or business unit.
EC Queue Instance	The system-generated queue instance ID given to this PeopleSoft business document.
EC Trans ID (EC transaction ID)	The transaction ID.
Incremented Sequence Nbr (incremented sequence number)	A system-generated sequence number. The EDI agent increments the number for each child row of a particular parent. It starts at 1 again when it reaches a new occurrence of the parent line.
Secondary Event Code	The secondary event code from the action code conversion.
Trading Partner Conversion	The trading partner ID associated with the customer, vendor, or business unit.

3. Move to the next field in the record definition and repeat step 2.

4. Scroll to the next source record definition and repeat steps 1 through 3.

5. Save the map.
6. Run the Outbound EDI Agent Preparer to create the Outbound EDI Agent SQC.

   After you create or modify outbound maps, you need to run a special compilation process that prepares the maps for use by the EDI agent.

   See Chapter 4, "Monitoring EDI Processing," page 53.

   Note. Run the Outbound EDI Agent Preparer only after you've made all your changes to outbound maps. You only need to run it only once to prepare all outbound maps.

See Also

Chapter 3, "Mapping EDI Transactions," Defining the Inbound PeopleSoft Business Document Format, page 32

Creating Data Mapping Profiles

This section provides an overview of data mapping profiles and discusses how to create data mapping profiles.

Understanding Data Mapping Profiles

When you add a trading partner, you assign to it a *map profile*, which lists the electronic commerce maps that the EDI agent can use to process transactions from the partner. The map profile serves two purposes:

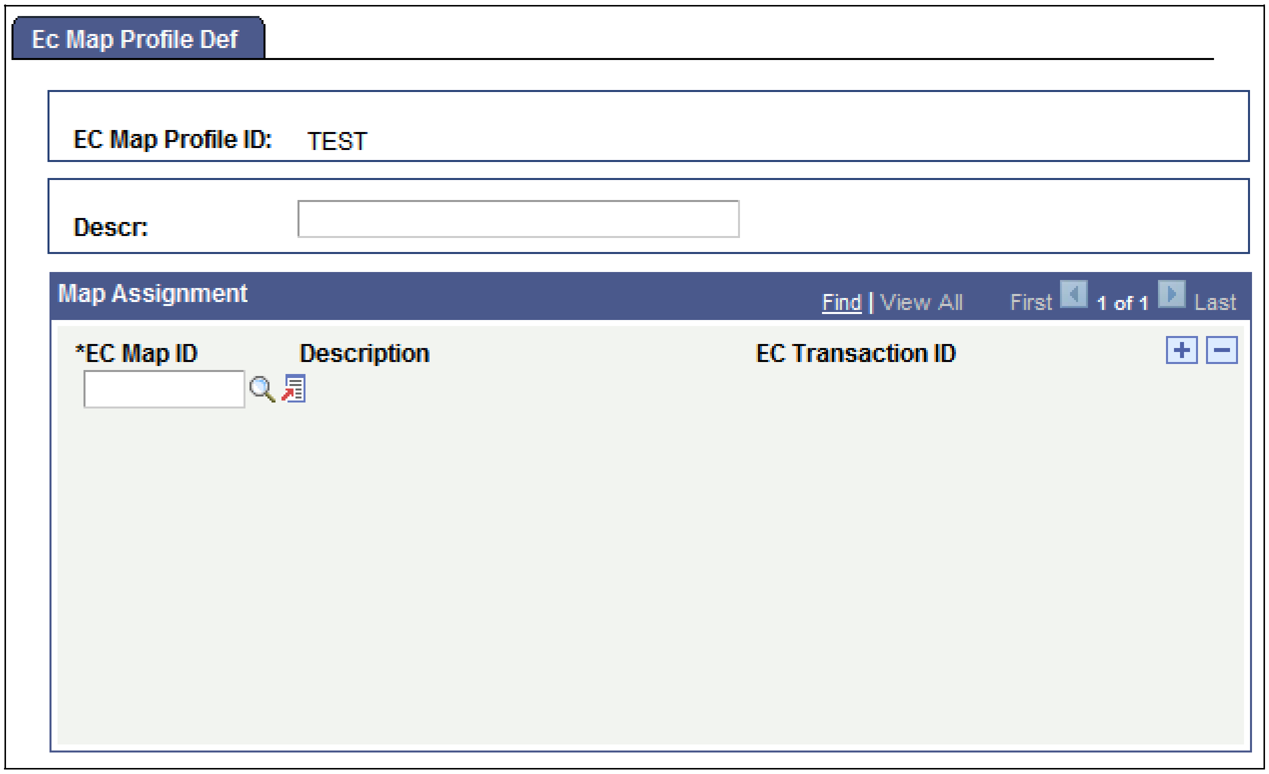
- It restricts a trading partner's access to transactions that they aren't authorized to exchange with your company.
- By assigning different map profiles to different trading partners, the EDI agent processes the same transaction differently for different partners.

Page Used to Create Data Mapping Profiles

Page Name	*Definition Name*	*Navigation*	*Usage*
Ec Map Profile Def (Map Profile Definition)	EC_MAP_PROFILE_DEF	PeopleTools, EDI Manager, Map EDI Transactions, Data Mapping Profiles	Create data mapping profiles.

Creating Data Mapping Profiles

Access the Ec Map Profile Def (Map Profile Definition) page.


Ec Map Profile Def

EC Map Profile ID: TEST

Descr:

Map Assignment

*EC Map ID	Description	EC Transaction ID

Ec Map Profile Def page

To create a map profile, add the maps for all the transactions that you want partners with this map profile to use. Select a map ID in the EC Map ID field, and then add a new row to add another map. Don't forget to add both inbound and outbound maps.

Using PeopleSoft EDI Manager for General Data Extraction

The primary purpose of PeopleSoft EDI Manager is to create PeopleSoft business documents, which contain business transaction data and are subsequently translated into X.12 or EDIFACT format and transmitted to a trading partner. However, you can also use it to extract data from database tables into a text file.

To create a text file from PeopleSoft database data:

1. Using Application Designer, create a view that extracts the data that you want and includes three EDI control fields.

   For example, to print a list of the countries in COUNTRY_TBL, create a view with the three EDI control fields described in the following table, plus the fields that you want from COUNTRY_TBL, perhaps Country and Description.

   The EDI control fields must be the first three fields in the view. The fields are:

Field	***Description***
ECTRANSID	The transaction ID for the transaction that you want to perform. You'll probably want to define a special transaction ID for this data extraction transaction.
ECQUEUEINSTANCE	The instance ID.
ECTRANSINOUTSW	Enter a value of *O* to specify an outgoing transaction.

   The Select statement for the view looks something like this:

```
SELECT
'CNTRY',
0,
'O',
COUNTRY,
DESCR
FROM
COUNTRY_TBL
```

   In this example, the transaction ID is *CNTRY,* the queue instance ID is always *0,* and the inbound/outbound switch is always *O* (outbound). Because the queue instance is the same for all records, the EDI agent processes them as a single transaction.

   Note. For this example to work, you must use PeopleSoft EDI Manager to define the transaction ID *CNTRY.*

2. Create a record definition that specifies the layout of the PeopleSoft business document file.

   This record definition typically consists of the ECFILEROWID field and all the fields from the table (for the example, COUNTRY and DESCR.

3. Define the outbound map.

   In the example, you would specify *CNTRY* as the transaction ID on the first page in the Outbound Map Definition component. On the second page, the source record is the view that you created in step 1 and the Where clause is:

   ```
   WHERE ECTRANSID = 'CNTRY'
   ```

   On the third page, the model file layout is the record definition that you created in step 2. After you've selected the record definition from the list, click the <Icon or Button Name> button to copy the field information into the page. You can use the default values for all the fields except ECFILEROWID. For ECFILEROWID, select Default in the Conversion Processing group box and enter whatever value you want to appear in the first column of the text file.

4. Run the Outbound EDI Agent Preparer process.

5. Add an entry to the ECQUEUE table.

   The EDI agent uses the ECQUEUE table to determine the transactions that are ready to be processed. The fields in the table are.

ECTRANSID	The transaction ID, which tells the EDI agent which map definition to use. In the example, the transaction ID is CNTRY.
ECQUEUEINSTANCE	The instance ID. The example uses a single instance ID, *0* (zero), for all the data.
ECTRANSINOUTSW	The inbound/outbound switch. Extracting data is an outbound transaction, so the value is *O* (the letter *O).*
ECBUSDOCID	The business document ID. Because you are not creating an EDI transaction, you can specify *0* (zero) in this field.
ECQUEUESTATUS	The status of the current record. The EDI agent processes records with the status *L.*

   The remaining fields in the table (BUSINESS_UNIT, ECENTITYCD_BU, ECCUSTVNDRVAL, and ECENTITYCD_EXT) relate to trading partner information. Because you are not creating an EDI transaction, you can use generic partner information. Use the default value GENR in all these fields.

6. Schedule the outbound EDI agent to run.

   The outbound EDI agent extracts your data and processes any other transactions on the ECQUEUE table with the status *L.*

   The status field ECQUEUESTATUS changes to *P* (Processed). To repeat the extraction, change the status back to *L.*

Chapter 4

Monitoring EDI Processing

This chapter provides an overview of Electric Data Interchange (EDI) agents and discusses how to:

- Prepare outbound maps for EDI agents.
- Schedule inbound EDI agents.
- Schedule outbound EDI agents.
- Review errors and summary data.

Note. You must run the outbound map preparation process before you can process any outgoing EDI transactions.

See Also

Chapter 3, "Mapping EDI Transactions," Defining Outbound Maps, page 41

Understanding EDI Agents

EDI agents copy EDI transaction data between PeopleSoft business documents and the PeopleSoft database, using the maps that you defined in the previous chapter. Two types of EDI agents are available:

- Inbound EDI agents, which process incoming transactions by copying data from PeopleSoft business documents into the database.
- Outbound EDI agents, which create PeopleSoft business documents from transaction data in the EDI staging tables.

If your system processes EDI transactions, you will usually want to have two EDI agents running on a regular schedule: one inbound agent and one outbound agent. You can also start other EDI agent instances for special purposes, such as processing the corrected version of a PeopleSoft business document that failed the first time.

Preparing Outbound Maps for EDI Agents

This section provides an overview of preparing outbound maps for EDI agents and discusses how to prepare outbound maps.

Understanding Preparing Outbound Maps for EDI Agents

When you create an outbound map definition, you define what data the EDI agent needs to extract from the staging tables. In order to select data from the tables, the EDI agent needs to run SQL statements whose details depend on the map definition.

Whenever you create or modify an outbound map definition, you need to run a preparation process that generates the appropriate SQL statements and compiles them into a format that the EDI agent can use. This process creates an SQC file.

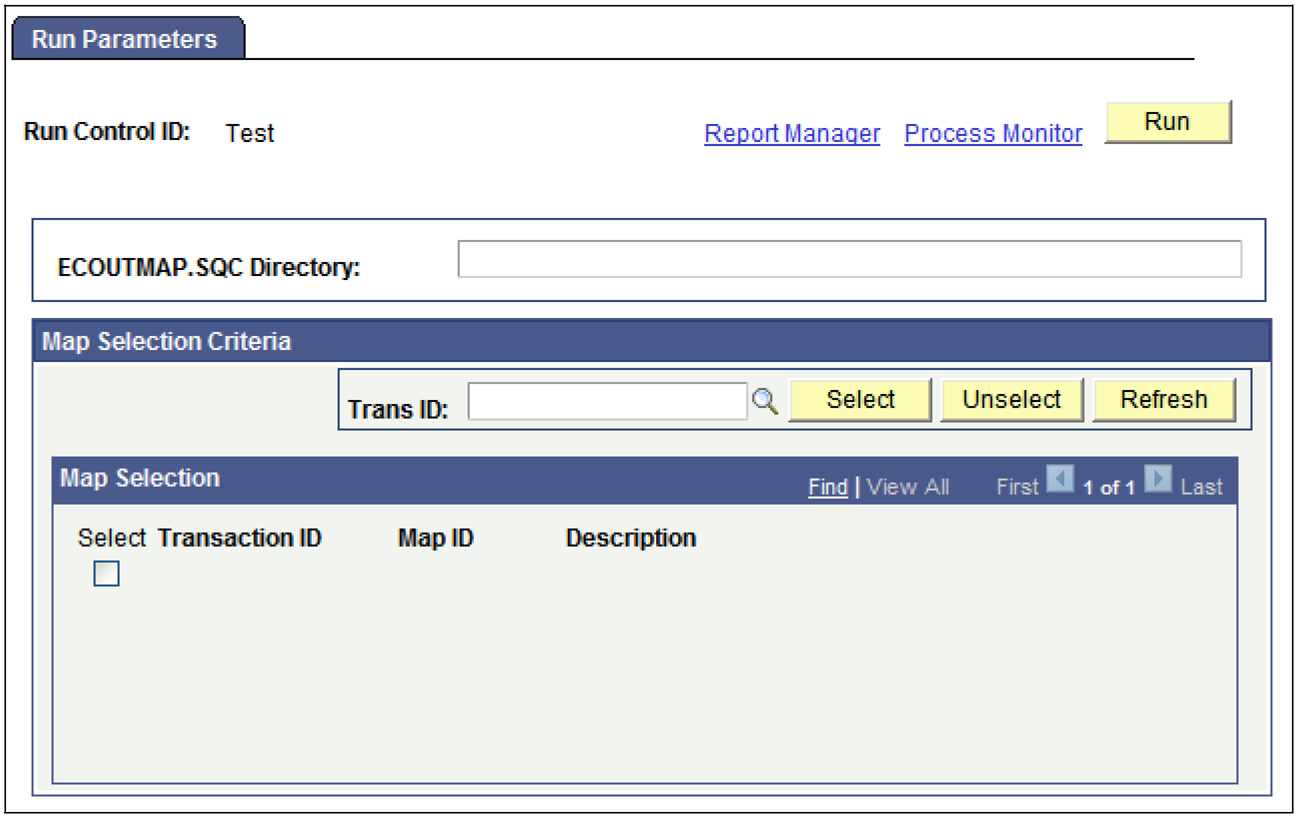
Note. You must run the Outbound EDI Agent Preparer process before you can process any outbound EDI transactions. PeopleSoft does not deliver a compiled version of the drivers; you have to compile them for your system.

Page Used to Prepare Outbound Maps for EDI Agents

Page Name	***Definition Name***	***Navigation***	***Usage***
Prepare Outbound Driver - Run Parameters	EC_RUN_OUTPREP_01	PeopleTools, EDI Manager, Monitor EDI Processing, Prepare Outbound Driver	Generates the appropriate SQL statements and compiles them into a format that the EDI agent can use for outbound maps.

Preparing Outbound Maps

Access the Prepare Outbound Driver - Run Parameters page.



Run Control ID: Test

ECOUTMAP.SQC Directory:

Trans ID:

Select	Transaction ID	Map ID	Description
☐			

Prepare Outbound Driver - Run Parameters page

To prepare outbound maps for the EDI agent:

1. Specify where to put the SQC that contains the outbound map definitions.

   In the ECOUTMAP.SQC Directory field, enter the directory where you want PeopleSoft Process Scheduler to put the SQC. The directory needs to be in the SQR search path of the workstation or server that will run the outbound EDI agent.

2. Click the Run button.

   The Process Scheduler Request page appears.

3. Specify the server where you want the SQR compiler to run.

4. Specify when you want the process to run.

   You will probably want to run the preparation process just once (then run it again after changing or adding maps).

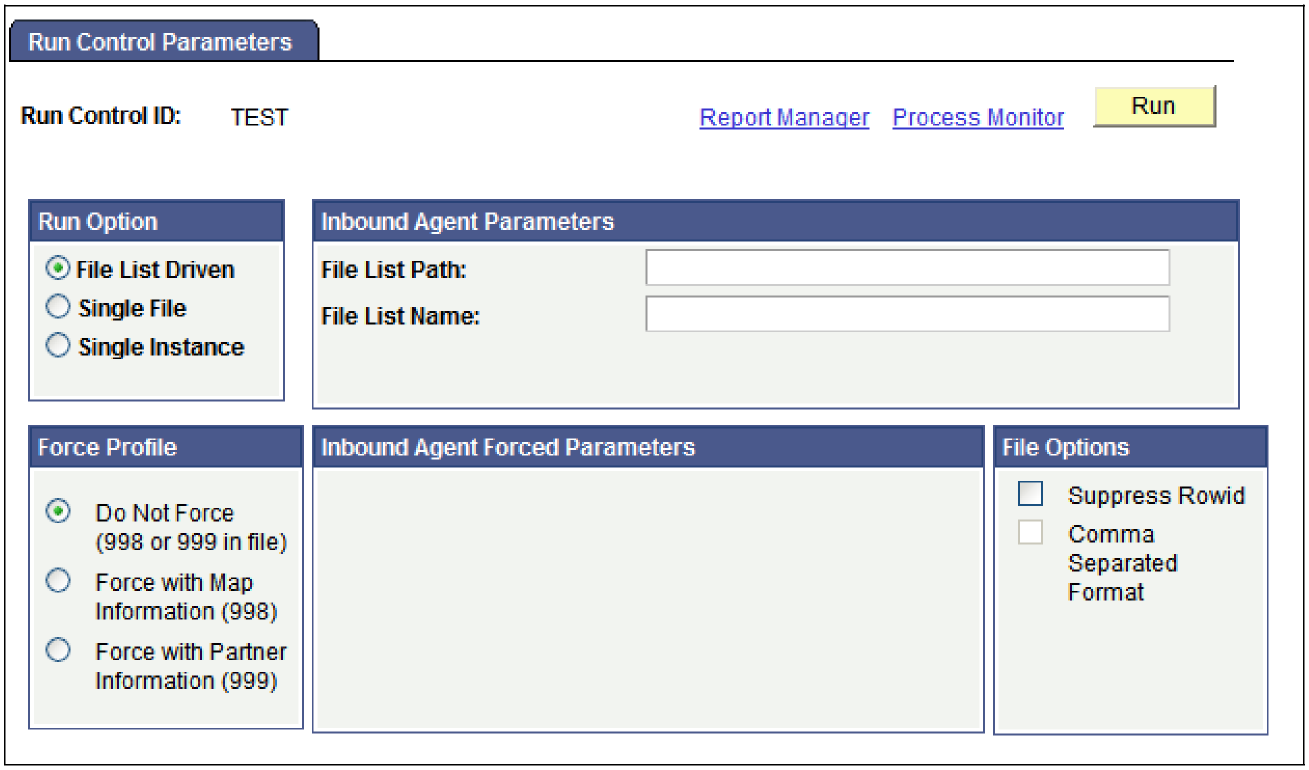
5. Click the OK button to run (or schedule to run) the outbound map preparation.

See Also

PeopleTools 8.52: PeopleSoft Process Scheduler, "Submitting and Scheduling Process Requests"

Scheduling Inbound EDI Agents

Select PeopleTools, EDI Manager, Monitor EDI Processing, Schedule Inbound EC Agent to access the Run Control Parameters page.


Run Control Parameters

Run Control ID: TEST

Run Option

[✓ File List Driven] Single File Single Instance

Inbound Agent Parameters

File List Path:	
File List Name:	

Force Profile

[✓ Do Not Force (998 or 999 in file)] Force with Map Information (998) Force with Partner Information (999)

Inbound Agent Forced Parameters

File Options

☐ Suppress Rowid
☐ Comma Separated Format

Run Control Parameters page

To schedule the inbound EDI agent to run:

1. Select the run option that the EDI agent uses to determine the documents or transactions to process.

File List Driven	Select to process all the PeopleSoft business documents whose filenames appear in a list file that the EDI translator software creates. The filenames must include their complete paths. Enter the directory that contains the PeopleSoft business document files and the list file in the File List Path field, and the filename of the list file in the File List Name field.
Single File	Select to process all the transactions in a specified PeopleSoft business document. Enter the directory that contains the file in the Single File Path field and the PeopleSoft business document file name in the Single File Name field.
Single Instance	Select to process a single transaction. Select the business document ID, transaction ID, and queue instance ID of the transaction that you want to process, in that order.

Note. The Single Instance option is available only for transactions that the EDI agent has already attempted to process. The transaction must have an EC queue instance ID.

Note. If the EDI agent runs on a server under MVS, it ignores the File List Path or Single File Path setting. Instead, it looks in the Job Control Language (JCL) for a DDNAME matching the specified file name. The File List Path option is not useful for MVS platforms because each file in the list must have a DDNAME..

2. Specify how the inbound agent determines which map definition to use.

   By default, the inbound agent looks for records with the row ID 999 to determine what type of transaction it is processing and, therefore, which map definition to use. However, you can also specify a transaction map or trading partner for all transactions that this inbound EDI agent processes, so the inbound EDI agent can process files that do not include control records.

   If you select the Do Not Force option, the inbound EDI agent looks for a control record at the beginning of each transaction in the incoming data file. You do not need to complete any other parameters.

   If you select Force with Map Information, the Inbound Agent Forced Parameters box displays lists from which to select the transaction map to use for all transactions in the incoming file.

   If you select Force with Partner Information, you can select the partner information to use for all transactions in the incoming file.

3. Select the file option that the EDI agent uses to display file data.

Suppress Rowid:	Select to map homogeneous files to the same table. A row ID of *000* must be specified in the map definition; however, the flat file need not have a record identifier.
Comma Separated Format:	Select to import files of variable length fields into PeopleSoft tables. This option is only available when the profile is forced with either 999 or 998 information.

Note. The separator should appear between each field in the file. The delimiter surrounds each field.

4. Click the Run button.

   The Process Scheduler Request page appears.

5. Specify on what server you want the agent to run.

Note. In a production workflow, you will usually want to run the EDI agent on a dedicated server.

6. Specify when and how often you want the agent to run.

   If you are running the EDI agent once, enter the date and time when you want to run it.

   If you are running the agents on a server, PeopleSoft Process Scheduler can run them once or periodically on a specified schedule.

Note. The Run Output option is not relevant for the EDI agent.

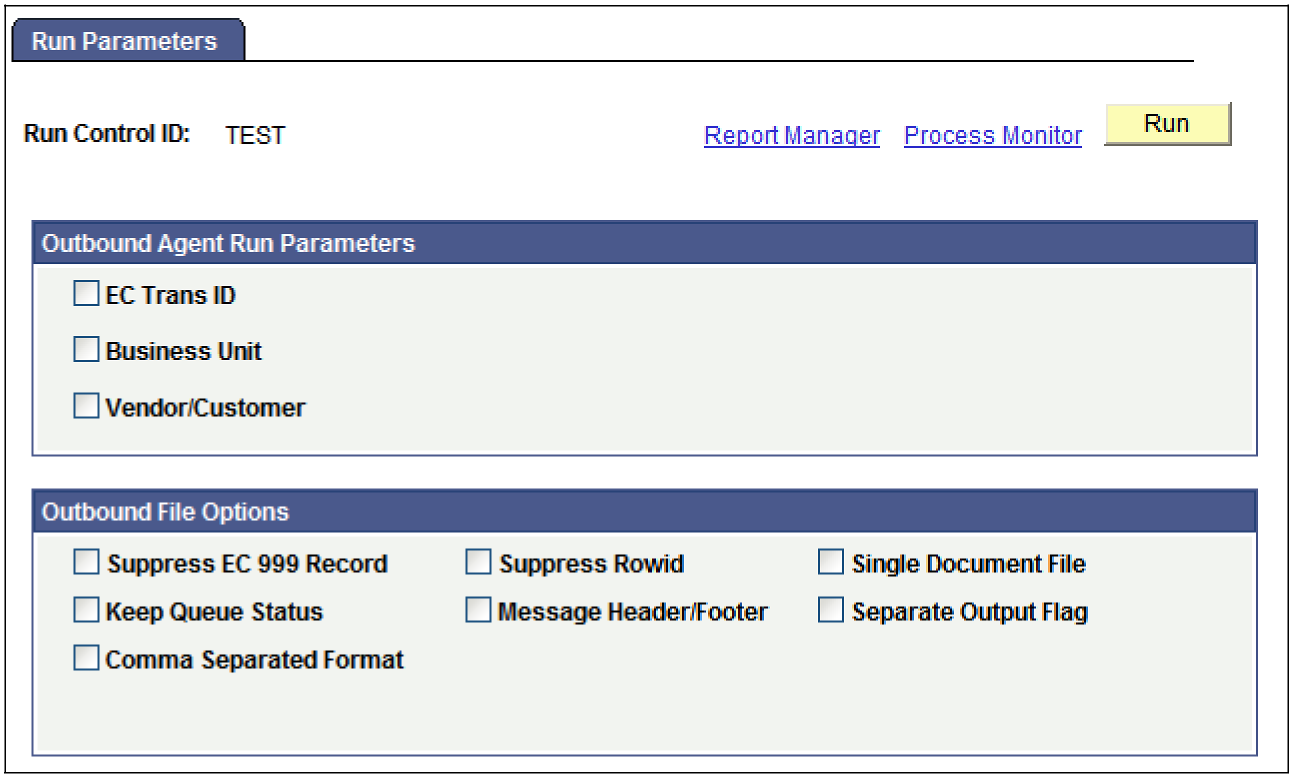
7. Click the OK button to run (or schedule to run) the EDI agent.

See Also

PeopleTools 8.52: PeopleSoft Process Scheduler, "Submitting and Scheduling Process Requests"

Scheduling Outbound EDI Agents

Select PeopleTools, EDI Manager, Monitor EDI Processing, Schedule Outbound EC Agent to access the Outbound EC Agent - Run Parameters page.


Run Parameters

Run Control ID: TEST

Report Manager Process Monitor Run

Outbound Agent Run Parameters

- ☐ EC Trans ID
- ☐ Business Unit
- ☐ Vendor/Customer

Outbound File Options

☐ Suppress EC 999 Record	☐ Suppress Rowid	☐ Single Document File
☐ Keep Queue Status	☐ Message Header/Footer	☐ Separate Output Flag
☐ Comma Separated Format		

Outbound EC Agent - Run Parameters page

To schedule the outbound EDI agent to run:

1. Specify the transactions that you want the EDI agent to process.

   To have the EDI agent process all pending transactions, clear all three check boxes. A pending transaction is one that an application extract process has added to the staging tables, and it has a status of *L* (Loaded).

   To process all transactions of a particular type, select the EC Trans ID check box, and then select the transaction ID from the drop-down list that appears.

   To process all transactions from a particular business unit, select the Business Unit check box and enter the business unit ID in the drop-down list that appears.

   To process all transactions for a particular trading partner, select the Vendor/Customer check box, and then enter the vendor ID, customer ID, or other internal ID for the desired partner.

   You can select multiple check boxes. For example, you could process all outgoing invoices for a particular vendor by selecting both the EC Trans ID and Vendor/Customer check boxes.

2. Specify how you want to format the outgoing transaction file.

   By default, the outbound EDI agent adds a record with the row ID *999* at the beginning of each transaction. This record identifies the transaction type in a way that is recognized by the PeopleSoft inbound EDI agent. If you select the Suppress EC 999 Record check box, the outbound EDI agent does not include this record in the outgoing file.

   If you do not want to include the row ID for each record in the outgoing file, select the Suppress Rowid check box.

   By default, the outbound EDI agent creates outbound transaction files in which the data for each field appears in the fixed column position specified in the outbound map definition. If you select the Comma Separated Format check box, the outbound EDI agent instead creates transaction files in which the fields are separated by a comma or other character rather than always appearing in a specified column position. The comma-separated option enables you to select the file options to read in flat file data that is comma-delimited.

   Note. The Comma Separated Format option is only available if you used forced agent mapping. Selecting the Suppress Rowid option enables you to load data without requiring a row ID. However, your map definition must contain a row ID even if you are suppressing it in the run control setup. The row ID must be *000.*

   The CSV separator is the character that the EDI agent uses to separate the fields in a record. The CSV delimiter is the character that the EDI agent uses to mark the beginning and end of a data value. In the following example, the separator is a comma, and the delimiter is a single quotation mark:

   'WOW','23.23','Big Deal'

   Note. When you use comma-separated format, the outbound EDI agent ignores the column positions specified in the outbound map definition. It adds data to the outbound file in the exact order in which the fields appear in the map definition. You must make sure that the order of rows map definition specifies the fields in the order that you want them to appear in the file.

   Select Single Document File to allow the results of the EDI to be written to a single output file. The business document ID is increased incrementally as though multiple documents were created. The file may contain multiple 999 records.

   Select Keep Queue Status to not update the ECQUEUE.ECQUEUESTATUS field at the completion of processing. This action allows the record in ECQUEUE to be processed multiple times.

   Select Message Header/Footer to process header and footer information before and following each business document. Header and footer information is stored with the trading partner definition.

3. Click the Run button.

4. Specify on which server you want the agent to run.

5. Specify when and how often you want the agent to run.

   If you are running the EDI agent once, enter the date and time when you want to run it.

   If you are running the agents on a server, Process Scheduler can run them once or periodically on a specified schedule.

6. Click the OK button to run, or schedule to run, the EDI agent.

See Also

PeopleTools 8.52: PeopleSoft Process Scheduler, "Submitting and Scheduling Process Requests"

Reviewing Errors and Summary Data

This section discusses how to:

- Review and correct errors.
- Review packages and transaction groups.

Reviewing and Correcting Errors

A predefined business process named Manage EDI, which is delivered with the application, routes information about any processing errors it encounters to the EDI Coordinator's worklist. The coordinator can select an item from the worklist to review the error.

To review and correct EDI processing errors:

1. Select PeopleTools, EDI Manager, View EDI Audit Trail, Business Document Errors.

   The Business Document Detail page appears. This page gives detailed information about the processing of each line in a PeopleSoft business document.

2. Review the processing details for the document.

   When you first open the page, the first row shows the processing information for the first record in the file, which is a 999 record. Browse the rows to view the other file records. The status of each row appears in the Status field.

3. (Optional) Correct the erroneous data in the PeopleSoft business document.

   Complete the remaining steps only if you correct errors.

   Note. Because this page enables you to change transaction data, you will want to carefully control access to it. In Operator Security, give only the EDI coordinator rights to edit this page.

4. Save the page.
5. Select PeopleTools, EDI Manager, View EDI Audit Trail, Transaction Maintenance.
6. Reset the EC queue status for the corrected transactions to *L*.

   When the EDI agent encounters an error while processing a transaction, it sets the status field in the pending transaction queue (ECQUEUE) to *E*. Resetting the status of a transaction to *L* makes the outbound EDI agent try to process it again.

Reviewing Packages and Transaction Groups

The EDI transaction set file that you receive over the network from an external trading partner might include any number of transactions bound for any number of internal trading partners. To manage all this transaction data, the file organizes it into several levels:

- The top level of organization is the *package level.* The package is the entire transaction set file, addressed to your company much as a mail package would be.
- The package can contain one or more *transaction groups.* Each transaction group is a set of transactions of the same type, with the same trading partners involved.
- Each transaction group includes one or more individual *units* of work. A unit of work is a single transaction that you commit or roll back as a whole.

When third-party-supplied EDI translation software creates PeopleSoft business documents, it divides the transaction set file at approximately the transaction group level. A PeopleSoft business document typically includes multiple units of work, and it may include multiple transaction groups, depending how the translation software is set up.

PeopleSoft business documents do not contain information about what packages or transaction groups their transactions come from. However, the EDI translation software can provide this information in a separate audit file, and the EDI agent can copy the information into audit tables in your database.

Note. For the auditing information to be available, the EDI translation software must write the package and transaction group information to an audit file, and the EDI agent must process this file using the predefined Audit inbound map.

To view summary information about a PeopleSoft business document:

1. Select PeopleTools, EDI Manager, View EDI Audit Trail, Business Document Summary.
2. Enter the business document ID of the document that you want to review.

   This page displays information about how the EDI agent processed the specified PeopleSoft business document. If the document included multiple transactions, browse the rows on the lower part of the page to see the information about each individual transaction.
3. Review the packages and transaction groups that this document is part of.

To view summary information about an EDI package:

1. Select PeopleTools, EDI Manager, View EDI Audit Trail, Package Log Summary.
2. Enter the package control ID for the package whose status you want to review.

   Browse the rows on the lower part of the page to view the transaction groups that were inside the package and the PeopleSoft business documents that were created from the data in the package.

To view summary information about an EDI transaction group:

1. Select PeopleTools, EDI Manager, View EDI Audit Trail, Transaction Group Log Summary.
2. Enter the transaction group control ID for the group whose status you want to review, or press Enter to select from a list of the groups.

Chapter 5

Using Forms Routing and the Outgoing Forms API

This chapter provides an overview of forms routing and discusses:

- Session-level operations.
- Query operations.
- Send operations.

Understanding Forms Routings

With PeopleSoft Application Designer, you can implement *routings*, which transfer data from one step in a business process to another. One type of routing is a *forms routing.* With a forms routing, the system takes data from a PeopleSoft page that the user is working on, enters it into a third-party form, and mails the completed form to designated users by means of the forms product's mail capabilities.

Note. You can also integrate electronic forms software so that users can send forms to PeopleSoft applications using PeopleSoft Component Interfaces.

To design a forms routing, you map the fields in a PeopleSoft page to the fields in the form that it wants to generate. When a user triggers the forms routing, the system copies the data from the page fields to the corresponding form fields, then sends the form.

See Also

PeopleTools 8.52: PeopleSoft Component Interfaces, "Understanding Component Interfaces"

PSFORMS.DLL

PeopleSoft applications communicate with the forms software using PSFORMS.DLL. To integrate a forms product with PeopleSoft applications, create a version of PSFORMS.DLL that supports the function calls that the PeopleSoft application uses to generate and send forms.

The function descriptions in this chapter describe the function calls that PeopleSoft applications make to PSFORMS.DLL and the responses that it expects to the calls.

Because PSFORMS.DLL should work at any installation, don't build in any site-specific information, such as the name of the forms server and database that the PeopleSoft applications access. For site-specific information, add entries to the Microsoft Windows registry. PSFORMS.DLL can read the entries to identify the forms server, database, and any other site-specific information that it needs.

Note. PeopleSoft supports any VIM- or MAPI-compliant email package. No further development is necessary.

Session-Level Operations

This section discusses the session-level operations in alphabetical order.

PsfCloseSession

Syntax

```
int FAR PASCAL PsfCloseSession(HSESSION hSession);
```

Description

Closes a session.

Parameters

Parameter	***Description***
hSession	The session handle assigned by PsfOpenSession when the application connected to PSFORMS.DLL.

Returns

This operation returns the following values:

Value	***Code***	***Meaning***
PSF_OK	0	The session was successfully closed.
PSF_NOSESS	4	*hSession* does not identify a current session.

PsfGetAPIInfo

Syntax

int FAR PASCAL **PsfGetAPIInfo**(LPPSFDEFNKEY *lpDefKey*);

Description

Provides the name of forms product that this DLL supports and returns the API version.

All PeopleSoft forms products use the same programming interface. They use the PsGetAPIInfo function to get the product name of the forms software that your version of PSFORMS.DLL supports.

PsfGetAPIInfo puts the product name as a string in the name field of the structure.

Parameters

Parameter	***Description***

lpDefKey	A pointer to a structure containing a character array for the function to which to write. The form of the structure is:
	`typedef struct`
	`{`
	`   char name[PSF_FIELDNAMELEN + 1];`
	`} PSFDEFNKEY;`

Returns

An integer identifying the version of this API that the DLL implements. The value is set in the PeopleSoft forms interface header file.

PsfOpenSession

Syntax

int FAR PASCAL **PsfOpenSession**(LPSTR *lpszUserName,* LPSTR *lpszPassword,* LPHSESSION *lphSession*);

Description

Connects the PeopleSoft system to the forms interface DLL using a specified login name and password.

PsfOpenSession logs in to the form software using the specified user name and password. It places a session handle in the buffer that *lphSession* points to. All subsequent API calls use this session handle.

PsfOpenSession always opens a new session, even when it could access a shared session.

Parameters

Parameter	***Description***
lpszUserName	A null-terminated string containing the user name for logging in to the forms software.
lpszPassword	A null-terminated string containing the password for the user identified by *lpszUserName.*
lphSession	A pointer to a buffer for the session handle that this function assigns.

Returns

This operation returns the following values:

Value	***Code***	***Meaning***
PSF_OK	0	The session was successfully opened.
PSF_NOSESS	4	The function was unable to establish a session.
PSF_NOSPEC	6	The forms database or server was not specified.

Query Operations

This section discusses the query operations in alphabetical order.

PsfGetFieldCount

Syntax

```
int FAR PASCAL PsfGetFieldCount(HSESSION hSession, LPPSFFORMDEFN lpFormData);
```

Description

PsfGetFieldCount gives the number of fields in a form by entering an integer in the wNumFields field of the *lpFormData* structure. The name of the desired form is in the Name field of the structure form.

The PeopleSoft system uses the field count to allocate an array large enough to hold all the fields. It passes a pointer to this array to PsfGetFieldList.

Parameters

Parameter	*Description*
hSession	The session handle assigned by PsfOpenSession.

lpFormData	A pointer to a structure that contains form information. The form of the structure is:
	`typedef struct`
	`{`
	`   PSFDEFNKEY formName;   // Name of the form`
	`   WORD wNumFields;  // Number of fields`
	`   char cSep;    // Separator character`
	`   LPPSFFIELDVAL lpFields;  // Pointer to field array`
	`} PSFFORMDEFN;`

Returns

This operation returns the following values:

Value	*Code*	*Meaning*
PSF_OK	0	The function ran successfully.
PSF_NOMAIL	2	Mail system failed.
PSF_NOFORM	3	The specified form is not accessible.
PSF_NODB	5	The forms database is not accessible.

PsfGetFieldList

Syntax

```
int FAR PASCAL PsfGetFieldList(HSESSION hSession, LPPFSFORMDEFN lpFormData);
```

Description

Provides a list of the fields on a form.

PsfGetFieldList provides the PeopleSoft application with information about the fields on a form. The name of the desired form is in the Name field of the structure form. The lpFields field of the structure points to the array of structures that PsfGetFieldList updates. Each item in the array has this structure:

```
typedef struct
{
   char szFldName[PSF_FORMNAMELEN + 1];    // Field name
   WORD wFldSize;    // Field size
   LPSTR lpszStrVal;    // Field value
} PSFFIELDVAL;
```

PsfGetFieldList enters data into the szFldName and wFldSize fields. The PeopleSoft system provides the field values before passing the structure to PsfSendForm.

PsfGetFieldList includes all fields required to send forms using the forms product, regardless of whether they appear on the user's form. For example, if a user's form contains a TO field but no CC field, then PsfGetFieldList includes both fields because a CC field is required to transmit a mail message. If the forms mail system supports and requires attributes such as sensitivity, return receipt, and priority, then PsfGetFieldList also returns these items as fields.

Parameters

Parameter	***Description***
hSession	The session handle assigned by PsfOpenSession when the PeopleSoft application connected to PSFORMS.DLL.

lpFormData	A pointer to a structure that contains form information. The form is:
	`typedef struct`
	`{`
	`   PSFDEFNKEY formName;   // Name of the form`
	`   WORD wNumFields;  // Number of fields`
	`   char cSep;    // Separator character`
	`   LPPSFFIELDVAL lpFields;  // Pointer to array`
	`} PSFFORMDEFN;`

Returns

This operation returns the following values:

Value	***Code***	***Meaning***
PSF_OK	0	The function ran successfully.
PSF_NOMAIL	2	Mail system failed.

Value	Code	Meaning
PSF_NOFORM	3	The form is not accessible.
PSF_NODB	5	The forms database is not accessible.

PsfGetFormCount

Syntax

```
int FAR PASCAL PsfGetFormCount(HSESSION hSession, LPINT lpnCount);
```

Description

Counts the available forms.

PsfGetFormCount sets *lpnCount* to a pointer to the number of forms available in the forms database. The PeopleSoft system uses the count to allocate an array large enough to hold all the forms. It passes a pointer to this array to PsfGetFormList.

Parameters

Parameter	Description
hSession	The session handle assigned by PsfOpenSession when PeopleSoft connected to PSFORMS.DLL.
lpnCount	A pointer to an integer that will receive the count.

Returns

This operation returns the following values:

Value	Code	Meaning
PSF_OK	0	The function ran successfully; *lpnCount* is set to the number of forms.
PSF_NOMAIL	2	Mail system failed.

Value	*Code*	*Meaning*
PSF_NODB	5	The forms database is not accessible.

PsfGetFormList

Syntax

```
int FAR PASCAL PsfGetFormList(HSESSION hSession, LPPFSFORMLIST lpFormList);
```

Description

Lists the available forms.

PsfGetFormList provides a list of the available forms in the forms database. The lpForms field of the structure points to the array of structures that PsfGetFieldList updates. Each item in the form name array has this structure:

```
typedef struct
{
   char name[PSF_FIELDNAMELEN + 1];
} PSFDEFNKEY;
```

Parameters

Parameter	*Description*
hSession	The session handle assigned by PsfOpenSession when the PeopleSoft application connected to PSFORMS.DLL.

lpFormList	A pointer to a structure to hold the list of form names. The form of the structure is:
	`typedef struct`
	`{`
	`   UINT nNumForms;  // The number of forms`
	`   LPPSFDEFNKEY lpForms;  // Form name array`
	`} PSFFORMLIST;`

Returns

This operation returns the following values:

Value	*Code*	*Meaning*
PSF_OK	0	The function ran successfully.

Value	***Code***	***Meaning***
PSF_NOMAIL	2	Mail system failed.
PSF_NODB	5	The forms database is not accessible.

PsfGetLastError

Syntax

```
int FAR PASCAL PsfGetLastError(HSESSION hSession, LPSTR lpszErrText, UINT
uBufSize, UINT uErrCode);
```

Description

PsfGetLastError gets the text of an error message based on an error code. The PeopleSoft system passes it the error code that it received and a pointer to a text buffer. PsfGetLastError copies the corresponding error text into the buffer, up to the size specified by *uBufSize*.

Parameters

Parameter	***Description***
hSession	The session handle assigned by PsfOpenSession when the PeopleSoft application connected to PSFORMS.DLL.
lpszErrText	A pointer to a buffer where the function places the error text.
uBufSize	The size of the buffer to which *lpszErrText* points.
uErrCode	The error code for which to retrieve the error text.

Returns

This operation returns the following values:

Value	***Code***	***Meaning***
PSF_OK	0	The function ran successfully.

Value	***Code***	***Meaning***
PSF_NOERR	9	No message was found for the specified error code.

Send Operations

This section discusses the send operation.

PsfSendForm

Syntax

`int FAR PASCAL` **`PsfSendForm`**`(HSESSION` *`hSession,`* `LPPSFFORMDEFN` *`lpFormData`*`);`

Description

Sends a form through the mail system with the specified field values.

PsfSendForm sends a form using the specified open session. The *lpFormData* structure has values in all fields, including the data values in the field array that it points to.

Parameters

Parameter	***Description***
hSession	The session handle assigned by PsfOpenSession when PeopleSoft connected to PSFORMS.DLL.

lpFormData	A pointer to a structure that contains form information. The form of the structure is:
	
	`typedef struct`
	`{`
	`   PSFDEFNKEYformName;   // Name of the form`
	`   WORD wNumFields; // Number of fields`
	`   charcSep;    // Separator character`
	`   LPPSFFIELDVAL lpFields;   // Pointer to array`
	`} PSFFORMDEFN;`

	Each item in the field array has this structure:
	
	`typedef struct`
	`{`
	`   char szFldName[PSF_FORMNAMELEN + 1]; // Field name`
	`   WORD wFldSize;    // Field size`
	`   LPSTR lpszStrVal;    // Value`
	`} PSFFIELDVAL;`

Returns

This operation returns the following values:

Value	***Code***	***Meaning***
PSF_OK	0	The function ran successfully.
PSF_NOFIELD	1	A field is missing or is the wrong size.
PSF_NOMAIL	2	Mail system failed.
PSF_NOFORM	3	The form is not accessible.
PSF_NODB	5	The forms database is not accessible.

Chapter 6

Using the Open Query ODBC Driver and API

This chapter provides an overview of PeopleSoft Open Query and discusses:

- Supported Open Database Connectivity (ODBC) functions.
- PeopleTools initialization procedures.
- Connection procedures.
- Information procedures.
- Catalog procedures (metadata).
- Using SQL execution procedures.
- SQL execution procedures.
- Execution models.
- Using retrieval procedures.
- Retrieval procedures.
- Status and error retrieval procedures.
- Transaction and connection termination procedures.
- ODBC compliance.
- ODBC to RDM data types.
- PeopleSoft Open Query ODBC API example.

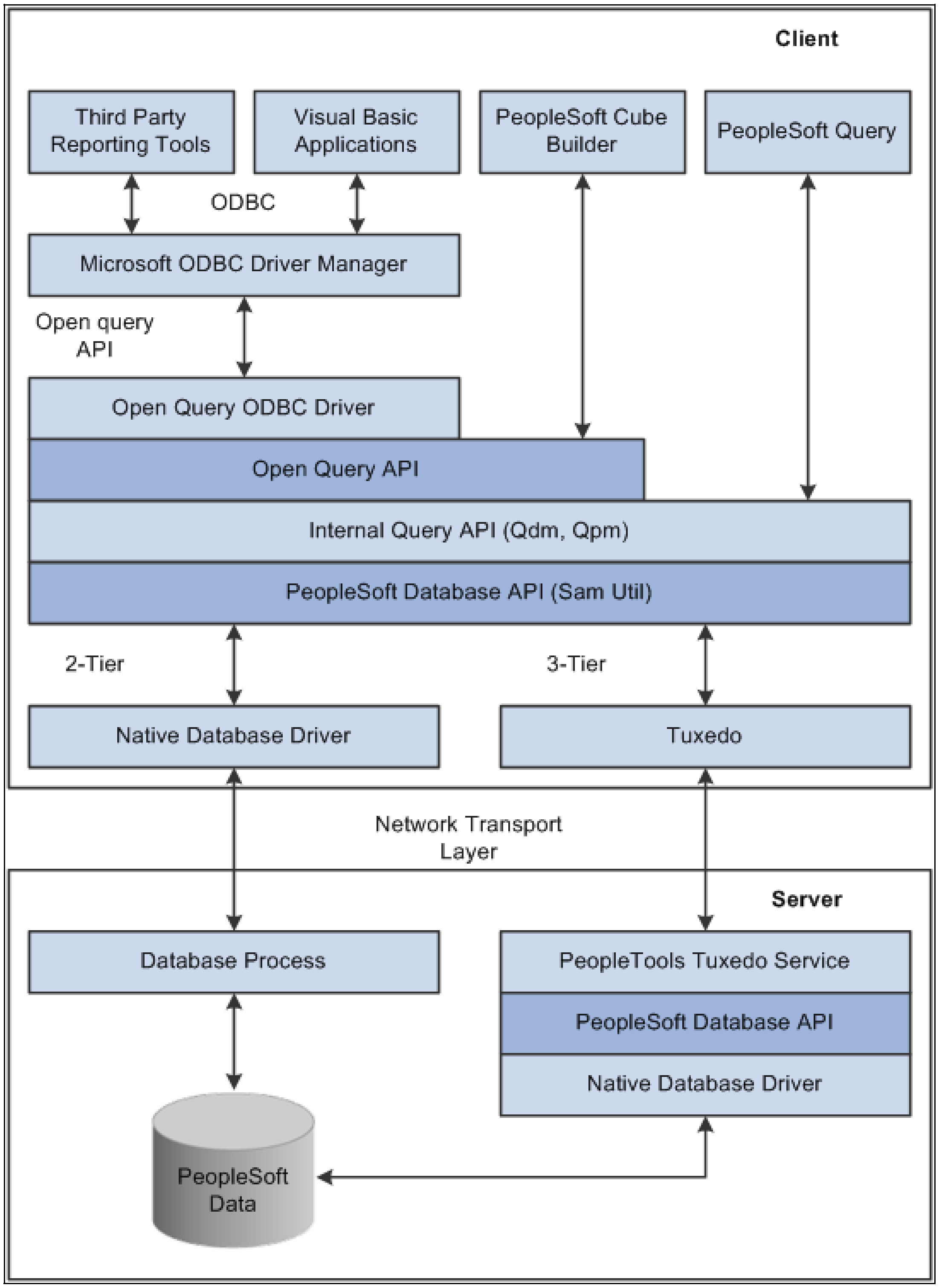
Understanding PeopleSoft Open Query

The PeopleSoft Open Query ODBC driver and API enable third-party reporting tools or applications to access PeopleSoft data in conformance with the PeopleSoft Query access architecture (the embedded SQL access intelligence provided by PeopleSoft Query). The PeopleSoft Query access architecture provides the following key features:

- Multiple levels of security
  - Query authorization
  - Operator security
  - Operator profile
  - Record level
  - Access group
  - Row level
  - Security record
- Standard query data access
  - Access to all supported PeopleSoft databases.
  - Ability to run stored PeopleSoft queries.
  - Automatic use of table sets.
  - Effective-dated output.
  - Translate values.
  - International translations.

Architecture

This diagram illustrates the components involved in PeopleSoft Open Query architecture. The blocks containing boldface type represent PeopleSoft Open Query components:



PeopleSoft Open Query architecture

This table describes PeopleSoft Open Query components:

Product	***Audience***	***Purpose***
Open Query ODBC	Third-party application vendors and implementation partners	Open, documented API providing access to PeopleSoft Query as a data source. Primary means of achieving programmatic interface to PeopleSoft data.
Open Query API	PeopleTools development	PeopleSoft proprietary. Exposes Query definition and runtime functions for use within PeopleTools. API used by ODBC driver and online analytical processing (OLAP).

PeopleSoft Open Query ODBC Driver

The PeopleSoft Open Query ODBC driver is the external means for extracting data from a PeopleSoft database.

Using ODBC as our external API has several advantages:

- Third-party tools do not need special information about PeopleSoft data.
- Most third-party reporting and query tools already support ODBC.
- Custom integration with PeopleTools is no longer required.
- Proprietary interface drivers (namely P2SPS.DLL) are eliminated.
- ODBC is supported by other application development tools, such as Visual Basic and PowerBuilder.
- Connectivity to PeopleSoft data is maintained by the PeopleSoft security architecture.
- Extension or modification of driver behavior is allowed by way of the ODBC standard.

The ODBC driver does not have any intrinsic knowledge of PeopleSoft data structures. Only the interface components exposed by way of PeopleSoft Open Query API are required to extract query data for a particular purpose.

This layer provides only the data modification and conversion code necessary to comply with the ODBC software development kit (SDK) standards. None of the PeopleTools structures are exposed at this level.

PeopleSoft Open Query API

The ODBC driver presents a stable interface to external applications. The PeopleSoft Open Query API is constantly under construction and therefore is not useful in this capacity. Because modifying behavior between releases presents a challenge when dealing with third-party applications, we created an intermediate layer between the ODBC driver and the underlying PeopleTools manager code. This wrapper code reduces the number of method calls to an acceptable level and allows the PeopleTools development team some leeway when new functionality is required or when the underlying code base is modified.

The PeopleSoft Open Query API enables the external driver (ODBC) and PeopleTools applications to focus solely on providing PeopleSoft data in the formats described by those products. It also abstracts the underlying connective architecture from the upper levels of the interface.

Because the architecture of this API is fluid, it is available only to PeopleTools development.

Note. You cannot use PeopleSoft Open Query APIs to write direct SQL, such as SELECT, UPDATE, INSERT, DELETE, and so on, against the database. You can use them only to execute stored procedures.

External Reporting Tools

The PeopleSoft Open Query ODBC driver enables third-party reporting tools or ODBC-enabled development applications to access a PeopleSoft database. The driver enforces user, table, and row-level security through internal PeopleTools APIs. A user can leverage PeopleSoft Query to easily create platform-independent queries that run against any supported PeopleSoft database platform.

Internet

The PeopleSoft Open Query API is a valuable link to all external data access mechanisms, including data over the internet.

Supported ODBC Functions

This quick-reference summary lists the ODBC API calls supported by the ODBC driver. API calls that are not supported return a SQL_ERROR. Each call is discussed in further detail in the following sections.

ODBC Call	***Function***
SQLAllocEnv	Allocates an environment handle for the ODBC connection.
SQLAllocConnect	Allocates a connection; returns a connection handle.
SQLAllocStmt	Allocates a statement handle for the specified connection.
SQLBindCol	Provides the buffer address for an answer column about to be fetched.
SQLBindParameter	Provides the value of a parameter (prompt variable) defined in the query.

ODBC Call	***Function***
SQLColAttributes	Returns result-column descriptor information for a result set.
SQLConnect	Connects to a PeopleSoft database.
SQLDescribeCol	Provides descriptors (data type and so on) for a result column.
SQLDescribeParam	Describes a parameter marker in a statement.
SQLDisconnect	Disconnects from the data source.
SQLDriverConnect	Connects to a PeopleSoft database, prompting the user for any login parameters not provided by the caller.
SQLError	Retrieves information about an error that occurred on a previous call.
SQLExecDirect	Prepares and executes a query. **Note.** Only the stored procedure syntax is supported.
SQLExecute	Executes a previously prepared query.
SQLFetch	Fetches a row of the answer set into the bound columns.
SQLFreeConnect	Closes the database connection and frees all resources that are associated with it.
SQLFreeStmt	Discards all resources that are associated with a previously prepared statement.
SQLGetData	Retrieves data for a specific column of the current fetched row. (Useful for long data, images, and so on.)
SQLGetFunctions	Tells applications which ODBC functions this driver supports.
SQLGetInfo	Retrieves information about the data source.

ODBC Call	***Function***
SQLGetRowCount	Returns the number of rows affected by the last execution.
SQLGetTypeInfo	Returns information about data types supported by the data source.
SQLNumParams	Returns the number of parameters in a statement.
SQLNumResultCols	Returns the number of result columns in the answer set of a prepared query.
SQLPrepare	Prepares a query for execution.
SQLProcedureColumns	Provides a list of queries and result columns available to the current operator and matching the specified qualifiers.
SQLProcedures	Retrieves a list of available stored procedures (queries).
SQLTransact	Commits or rolls back the current transaction.

The ODBC functions in the following table are supported calls with no underlying functionality. These functions exist to ensure compatibility with ODBC applications:

ODBC Call	***Function***
SQLColumns	Retrieves column information from the database.
SQLForeignKeys	Retrieves database information concerning foreign keys.
SQLGetConnectOption	Gets connection option information.
SQLGetCursorName	Gets the name of the cursor.
SQLGetStmtOption	Gets statement option information.
SQLMoreResults	Returns whether or not another result set is pending.
SQLPrimaryKeys	Retrieves database information on primary keys.

ODBC Call	***Function***
SQLSetConnectOption	Sets database connection options.
SQLSetCursorName	Sets the name of the cursor to be used with the statement.
SQLSetScrollOptions	Sets options to control cursor scrolling.
SQLSetStmtOption	Sets options for the statement.
SQLSpecialColumns	Retrieves information about optimal keys or automatically incremented columns.
SQLStatistics	Retrieves statistics in tables and indices from the database.
SQLTables	Retrieves a list of tables or views in the database.

PeopleTools Initialization Procedures

This section discusses the PeopleTools initialization procedures in alphabetical order.

SQLAllocEnv

Syntax

RETCODE **SQLAllocEnv***(phenv)*

Description

SQLAllocEnv allocates memory for an environment handle and initializes the ODBC call level interface for use by an application.

Parameters

This table describes the parameter:

Argument	*Type*	*Use*	*Description*
phenv	HENV FAR *	Output	Pointer to storage for the environment handle.

SQLAllocConnect

Syntax

`RETCODE` **`SQLAllocConnect`**`(`*`henv,phdbc`*`)`

Description

SQLAllocConnect allocates memory for a connection handle within the environment, identified by *henv*. This is called after SQLAllocEnv.

Parameters

This table describes the parameters:

Argument	*Type*	*Use*	*Description*
henv	HENV	Input	Environment handle.
phdbc	HDBC FAR *	Output	Pointer to storage for the connection handle.

SQLFreeEnv

Syntax

`RETCODE` **`SQLFreeEnv`**`(`*`henv`*`)`

Description

SQLFreeEnv releases an environment handle and frees all memory that is associated with the handle. This is called after SQLFreeConnect.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
henv	HENV	Input	Environment handle.

Connection Procedures

This section discusses the PeopleTools connection procedures in alphabetical order.

ODBC uses an abstraction that maps a single name (called the data source name or DSN) to all the necessary underlying software components required to access the data. The data source name is chosen by an end user or a system administrator and should express the kind of data that the DSN represents. ODBC data source mapping information is maintained in the registry in Microsoft Windows.

In order to connect to a PeopleSoft data source, several pieces of information are needed. With the introduction in PeopleTools of multiple sign-on capabilities, users must be prompted for required log in information. By default, PeopleTools installs an ODBC data source with the name PeopleSoft PeopleTools. This DSN has no references to PeopleSoft connection information. It is in effect an empty data source. Using this data source forces the underlying PeopleSoft security mechanisms to present the user with a sign-on dialog box. The user completes this as for a normal PeopleSoft application.

As per the ODBC standard, the PeopleSoft driver enables the user to create a data source that provides the information necessary to log in. If the information matches a current session, the user is not prompted to log in again.

The connection environment is affected by the workstation settings for PeopleSoft Process Scheduler. The values for the DBBIN and TOOLBIN are searched for the necessary support files required to log in to a database. Hence these values need to be valid. In the case of a three-tier logon, the value for DBBIN should be set empty.

Note. ODBC supports three connection functions: SQLConnect, SQLDriverConnect, and SQLBrowseConnect. SQLBrowseConnect is not supported by the PeopleSoft Open Query ODBC driver.

SQLConnect

Syntax

RETCODE **SQLConnect***(hdbc,szDSN,cbDSN,szUID,cbUID,szAuthStr,cbAuthStr)*

Description

SQLConnect loads a driver and establishes a connection to a data source. The connection handle references storage of all information about the connection, including status, transaction state, and error information.

This function assumes that a connection may be completed by supplying only a DSN, user, and password. It further assumes that the application will either prompt the end user for security information, the security information is hard-coded, or the security information can be obtained from a centralized security server, such as Kerberos.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.
szDSN	UCHAR FAR*	Input	Data source name.
cbDSN	SWORD	Input	Length of *szDSN*
szUID	UCHAR FAR*	Input	User identifier.
cbUID	SWORD	Input	Length of *szUID.*
szAuthStr	UCHAR FAR*	Input	Authentication string (typically the password).
cbAuthStr	SWORD	Input	Length of *szAuthStr.*

SQLDriverConnect

Syntax

RETCODE **SQLDriverConnect***(hdbc,hwnd,szConnStrIn,cbConnStrIn,szConnStrOut, cbConnStrOutMax,pcbConnStrOut,fDriverCompletion)*

Description

SQLDriverConnect handles the entire connection process for an application, prompting the end user for any information to complete the connection. The programmer can specify as much or as little about the connection as necessary. The following example illustrates the simplest case, in which the application does not specify any information at all about the connection. It supplies the connection handle returned from SQLAllocConnect, a window handle, and option specification of SQL_DRIVER_COMPLETE, and zero or NULL for the rest of the arguments.

```
rc = SQLDriverConnect(hdbc, hwnd, NULL, 0, NULL, 0, 0, SQL_DRIVER_COMPLETE);
```

The following keywords are used and supported by the PeopleSoft ODBC driver:

- DSN: Data source name required by ODBC.
- APPNAME: Application server name used for three-tier logon only.
- DBTYPE: Database type of login can be any of the following values:
    - DB2: DB2 using Centura SQL network.
    - DB2400: DB2 on AS/400 using client access.
    - DB2ODBC: DB2 using the IBM ODBC driver.
    - DB2UNIX: DB2 UNIX driver.
    - ORACLE: Oracle using the OCI interface.
    - INFORMIX: Native Informix.
    - SYBASE: Native Sybase.
    - MICROSFT: SQL Server using ODBC.
    - APPSRV: Used to indicate that the database name is actually an application server name.
- DBNAME: Name of the database or alias.
- DBQ: Used to combine values separated by a slash *(/),* such as APPNAME/DBTYPE/DBNAME. The APPNAME value and the following slash are dropped when not in three-tier.
- SERVER: Name of the database server, used by Sybase and Informix.
- UID: PeopleSoft user ID.
- PWD: Password associated with the PeopleSoft user.

The driver uses any information it retrieves from the ODBC.INI file or registry to augment the information passed to it in the connection string. If the information in the ODBC.INI file or registry duplicates information in the connection string, the driver uses the information in the connection string.

The existing PeopleSoft database connection dialog box prompts the user for any required connection information.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.
hwnd	HWND	Input	Window handle. The application can pass the handle of the parent window, if applicable, or a null pointer if either the window handle is not applicable or if SQLDriverConnect does not present any dialog boxes.
szConnStrIn	UCHAR FAR*	Input	A full connection string, a partial connection string, or an empty string.
cbConnStrIn	SWORD	Input	Length of szConnStrIn.
szConnStrOut	UCHAR FAR*	Output	Pointer to storage for the completed connection string. Upon successful connection to the target data source, this buffer contains the completed connection string. Applications should allocate at least 255 bytes for this buffer.
cbConnStrOutMax	SWORD	Input	Maximum length of the szConnStrOut buffer.
pcbConnStrOut	SWORD FAR*	Output	Pointer to the total number of bytes returned in szConnStrOut. If the number of bytes is greater than or equal to cbConnStrOutMax, the completed connection string in szConnStrOut is truncated to cbConnStrOutMax -1.

Argument	***Type***	***Use***	***Description***
fDriverCompletion	UWORD	Input	Flag that indicates whether the Driver Manager or the driver must prompt for more connection information: SQL_DRIVER_PROMPT, SQL_DRIVER_COMPLETE, SQL_DRIVER_COMPLETE_REQUIRED, or SQL_DRIVER_NOPROMPT.

Information Procedures

ODBC defines these functions as a means for the application to get information about the ODBC driver and data source. Typically, the application calls these functions passing a value of the particular type of information of interest. These values are numerous and are defined in the Microsoft ODBC SDK reference manual.

SQLGetInfo

Syntax

```
RETCODE SQLGetInfo(hdbc,fInfoType,rgbInfoValue,cbInfoValueMax,pcbInfoValue)
```

Description

SQLGetInfo returns general information about the driver and data source that is associated with a connection handle.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.

Argument	***Type***	***Use***	***Description***
fInfoType	UWORD	Input	Type of information. *FInfoType* must be a value representing the type of interest.
rgbInfoValue	PTR	Output	Pointer to storage for the information. Depends on the *fInfoType* requested.
cbInfoValueMax	SWORD	Input	Maximum length of the *rgbInfoValue* buffer.
pcbInfoValue	SWORD FAR *	Output	The total number of bytes (excluding the null termination byte for character data) available to return in *rgbInfoValue.*

SQLFunctions

Syntax

`RETCODE` **`SQLGetFunctions`***`(hdbc,fFunction,pfExists))`*

Description

SQLGetFunctions returns information about whether a driver supports a specific ODBC function.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.
fFunction	UWORD	Input	SQL_API_ALL_FUNCTIONS or a #define value that identifies the ODBC function of interest.

Argument	***Type***	***Use***	***Description***
pfExists	UWORD FAR *	Output	If *fFunction* is SQL_API_ALL_FUNCTIONS, *pfExists* points to a UWORD array with 100 elements. The array is indexed by #define values that are used by *fFunction* to identify each ODBC function; some elements of the array are unused and reserved for future use. An element has a value of *True* if it identifies an ODBC function that is supported by the driver. It has a value of *False* if it identifies an ODBC function that is not supported by the driver or does not identify an ODBC function.

SQLGetTypeInfo

Syntax

RETCODE **SQLGetTypeInfo***(hstmt,fSqlType)*

Description

SQLGetTypeInfo returns information about data types that are supported by the data source. The driver returns the information in the form of a SQL result set.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle for the result set.
fSqlType	SWORD	Input	The SQL data type.

Catalog Procedures (Metadata)

ODBC listing procedures supply the client with catalog table information. The PeopleSoft ODBC driver supports listings of queries and columns using PeopleSoft metadata.

SQLProcedures

Syntax

RETCODE **SQLProcedures***(hstmt,szProcQualifier,cbProcQualifier,szProcOwner, cbProcOwner,szProcName,cbProcName)*

Description

SQLProcedures returns the list of procedure names that are stored in a specific data source. *Procedure* is a generic term used to describe executable objects or named entities that can be invoked using input and output parameters and that can return result sets similar to the results returned by SQL Select statements.

This function is typically used before statement execution to retrieve information about procedures available from the data source catalog.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
szProcQualifier	UCHAR FAR *	Input	Procedure qualifier.
cbProcQualifier	SWORD	Input	Length of *szProcQualifier*..
szProcOwner	UCHAR FAR *	Input	String search pattern for procedure owner names.
cbProcOwner	SWORD	Input	Length of *szProcOwner*.
szProcName	UCHAR FAR *	Input	String search pattern for procedure names.

Argument	***Type***	***Use***	***Description***
cbProcName	SWORD	Input	Length of *szProcName.*

SQLProcedures returns the results as a standard result set (when SQLFetch is called), ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME, PROCEDURE_REMARKS, and PROCEDURE_TYPE.

This table lists the columns that are in the PeopleSoft result set:

Column Name	***Data Type***	***Description***
PROCEDURE_QUALIFIER	SQL_CHAR(128)	''
PROCEDURE_OWNER	SQL_CHAR(128)	'QUERY'
PROCEDURE_NAME	SQL_CHAR(128)	Query name with punctuation and spaces converted to underscore
REMARKS	SQL_CHAR(256)	Description of the Query, currently unused
PROCEDURE_TYPE	SQL_INT	SQL_PT_PROCEDURE

SQLProcedureColumns

Syntax

RETCODE **SQLProcedureColumns***(hstmt,szProcQualifier,cbProcQualifier,szProcOwner, cbProcOwner,szProcName,cbProcName,szColumnName,cbColumnName)*

Description

SQLProcedureColumns returns a list of input and output parameters, as well as the columns that make up the result set for the specified procedures. The driver returns the information as a result set on the specified statement handle.

This function is typically used before statement execution to retrieve information about procedure parameters and columns from the data source's catalog.

The PeopleSoft driver returns information for the first query that is requested only. It does not return results for multiple queries. The driver will the new query API functions QpmDescribeParm and QpmDescribeCol. QpmDescribeParm walks the query definition that is stored in hstmt and for each prompt variable returns a SQLProcedureColumns result row of COLUMN_TYPE equal to SQL_PARM_INPUT. QpmDescribeParm walks the same query definition and for each result column returns a SQLProcedureColumns result row of COLUMN_TYPE equal to SQL_RESULT_COL. The szProcQualifier and szProcOwner criteria are ignored. The result set returned is for the current user ID. The result set columns for Procedure Qualifier and Procedure Remarks do not apply and are set to NULL with a one-byte column length. The Procedure Owner column is set to either the user ID or Public.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle
szProcQualifier	UCHAR FAR*	Input	Procedure qualifier name
cbProcQualifier	SWORD	Input	Length of szProcQualifier
szProcOwner	UCHAR FAR*	Input	String search pattern for procedure owner names
cbProcOwner	SWORD	Input	Length of szProcOwner
szProcName	UCHAR FAR*	Input	String search pattern for procedure names
cbProcName	SWORD	Input	Length of szProcName
szColumnName	UCHAR FAR*	Input	String search pattern for column names
cbColumnName	SWORD	Input	Length of szColumnName

SQLProcedureColumns returns the results as a standard result set (when SQLFetch is called), ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCDURE_NAME, and COLUMN_TYPE.

This table lists the columns in the result set:

Column Name	***Data Type***	***Description***
PROCEDURE_QUALIFIER	SQL_CHAR(128)	N/A
PROCEDURE_OWNER	SQL_CHAR(128)	N/A
PROCEDURE_NAME	SQL_CHAR(128)	Procedure identifier
COLUMN_NAME	SQL_CHAR(128)	Procedure column identifier
COLUMN_TYPE	SQL_INT	SQL_PARAM_INPUT or SQL_RESULT_COL
DATA_TYPE	SQL_INT	SQL data type
TYPE_NAME	SQL_CHAR(128)	Data type name of procedure column
PRECISION	SQL_INT	Precision of the procedure column
LENGTH	SQL_INT	Length in bytes of data transferred on a SQLGetData or SQLFetch operation
SCALE	SQL_INT	Scale of procedure column
RADIX	SQL_INT	N/A
NULLABLE	SQL_INT	Whether the procedure column accepts a NULL value
REMARKS	SQL_CHAR(256)	A description of the procedure column

Using SQL Execution Procedures

The minimum ODBC SQL conformance level requires the driver to support:

- Data Definition Language (DDL): Create Table and Drop Table
- Data Manipulation Language (DML): simple Select, Insert, Update Searched, and Delete Searched.

- Expressions: simple (such as A > B + C).
- Data types: Char, Varchar, or Long Varchar.

The PeopleSoft Open Query ODBC driver does not support the minimum SQL conformance level even though it reports supporting extended syntax. PeopleSoft Open Query supports only the ODBC extended SQL grammar for stored procedures. The stored procedure syntax is:

```
{[? = ] call procedure_name [ (param, ...)]}
```

The stored procedure execution model supports the independent creation of a SQL statement. Independent creation is done through PeopleSoft Query. However, instead of a stored procedure, the result is a PeopleSoft Query object.

A *statement handle* references statement information, such as network information, SQLSTATE values and error messages, cursor names, number of result set columns, and status information for SQL statement processing. Before an application can execute a SQL statement, it must allocate a statement handle for the statement. To allocate a statement handle, an application declares a variable of type hstmt and passes its address to SQLAllocStmt.

See Also

Chapter 6, "Using the Open Query ODBC Driver and API," Execution Models, page 106

SQL Execution Procedures

This section discusses the SQL execution procedures in alphabetical order.

SQLAllocStmt

Syntax

```
RETCODE SQLAllocStmt(hdbc,phstmt)
```

Description

SQLAllocStmt allocates memory for a statement handle and associates the statement handle with the connection that is specified by the connection handle.

If the application calls SQLAllocStmt with a pointer to a valid statement handle, the driver overwrites the statement handle without regard to its previous contents.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.
phstmt	HSTMT FAR*	Output	Pointer to storage for the statement handle.

SQLExecDirect

Syntax

```
RETCODE SQLExecDirect(hstmt,szSqlStr,cbSqlStr)
```

Description

SQLExecDirect executes a preparable statement, using the current values of the parameter marker variables if any parameters exist in the statement. The application calls SQLExecDirect to send a SQL statement to the data source. The driver modifies the statement to use the form of SQL used by the data source, then submits it to the data source. In particular, the driver modifies the escape clauses that are used to define ODBC-specific SQL grammar extensions.

The application can include one or more parameter markers in the SQL statement. To include a parameter marker, the application embeds a question mark (?) into the SQL statement at the appropriate position. It is unnecessary to use any parameter markers, as PeopleSoft Query objects know the exact number of prompt values. The PeopleSoft driver prompts the user for input values if no values were supplied through the input or the SQLBindParameter function.

Only stored procedures (predefined queries) are supported.

In addition to the ODBC error conditions, the PeopleSoft driver returns an error condition if the following conditions are true:

- A valid PeopleSoft query name cannot be found or loaded.
- Prompt values cannot be satisfied using a prompting page.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle

Argument	***Type***	***Use***	***Description***
szSqlStr	UCHAR FAR*	Input	SQL statement to be executed.
cbSqlStr	SDWORD	Input	Length of *szSqlStr*.

SQLPrepare

Syntax

RETCODE **SQLPrepare***(hstmt,szSqlStr,cbSqlStr)*

Description

SQLPrepare prepares a SQL string for execution. The application calls SQLPrepare to send a SQL statement to the data source for preparation. The application can include one or more parameter markers in the SQL statement. To include a parameter marker, the application embeds a question mark (?) into the SQL string at the appropriate position. After a statement is prepared, the application uses hstmt to refer to the statement in later function calls. The prepared statement that is associated with the hstmt may be executed again by calling SQLExecute until the application frees the hstmt with a call to SQLFreeStmt with the SQL_DROP option or until the hstmt is used in a call to SQLPrepare, SQLExecDirect, or one of the catalog functions (SQLColumns, SQLTables, and so on). After the application prepares a statement, it can request information about the format of the result set.

Only stored procedures (predefined queries) are supported.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
szSqlStr	UCHAR FAR*	Input	SQL statement to be executed.
cbSqlStr	SDWORD	Input	Length of *szSqlStr*.

SQLExecute

Syntax

RETCODE **SQLExecute***(hstmt)*

Description

SQLExecute executes a prepared statement, using the current values of the parameter marker variables if any parameter markers exist in the statement. SQLExecute executes a statement prepared by SQLPrepare. After the application processes or discards the results from a call to SQLExecute, the application can call SQLExecute again with new parameter values.

To execute a Select statement more than once, the application must call SQLFreeStmt with the SQL_CLOSE parameter before reissuing the Select statement.

As in the SQLExecDirect function, the PeopleSoft ODBC driver prompts the user for input values if they have not been supplied.

Note. For applications to be flexible enough for using SQL for a particular purpose, you must provide the application a means to query the ODBC driver for information pertaining to required storage and data types. This is done using descriptive functions defined by the ODBC specification. ODBC-enabled applications use these functions to dynamically query the driver for information about the result set and the input and output values.

Parameters

This table describes the parameter:

Argument	*Type*	*Use*	*Description*
hstmt	HSTMT	Input	Statement handle.

SQLColAttributes

Syntax

RETCODE **SQLColAttributes***(hstmt,icol,fDescType,rbgDesc,cbValueMax,pcbValue)*

Description

SQLColAttributes returns descriptor information for a column in a result set; it cannot be used to return information about the bookmark column (column 0). Descriptor information is returned as a character string, a 32-bit descriptor-dependent value, or an integer value.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
icol	UWORD	Input	Column number of result data.
fDescType	UWORD	Input	Valid descriptor type.
rbgDesc	PTR	Output	Pointer to storage for descriptor information.
cbValueMax	SWORD	Input	Maximum buffer size.
pcbValue	SWORD FAR*	Output	Output length of data in buffer.

SQLDescribeCol

Syntax

```
RETCODE SQLDescribeCol(hstmt,icol,szColName,cbColNameMax,pcbColName,pfSqlType,
pcbColDef,pibScale,pfNullable)
```

Description

SQLDescribeCol returns the result descriptor, column name, type, precision, scale, and nullability for one column in the result set. An application typically calls SQLDescribeCol after a call to SQLPrepare and before or after the associated call to SQLExecute. An application can also call SQLDescribeCol after a call to SQLExecDirect.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
icol	UWORD	Input	Column number of result data.
szColName	UCHAR FAR*	Output	Pointer to storage for the column name.
cbColNameMax	SWORD	Input	Maximum length of the szColName buffer.
pcbColName	SWORD FAR*	Output	Total number of bytes available to return in *szColName.*
pfSqlType	SWORD FAR*	Output	The SQL data type of the column.
pcbColDef	UDWORD FAR*	Output	The precision of the column on the data source.
pibScale	SWORD FAR*	Output	The scale of the column on the data source.
pfNullable	SWORD FAR*	Output	Indicates whether the column allows NULL values.

SQLDescribeParam

Syntax

```
RETCODE SQLDescribeParam(hstmt,ipar,pfSqlType,pcbColDef,pibScale,pfNullable)
```

Description

SQLDescribeParam returns the description of a parameter marker that is associated with a prepared SQL statement. In terms of PeopleSoft Query objects, this is the description of the prompt values required to fulfill the query keys.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
ipar	UWORD	Input	Marker number.
pfSqlType	SWORD FAR*	Output	Pointer to storage for the SQL type.
pcbColDef	SWORD FAR*	Output	Pointer to storage for precision of value.
pibScale	SWORD FAR*	Output	Pointer to storage for scale of value.
pfNullable	UDWORD FAR*	Output	Pointer to storage for nullable flag.

SQLGetRowCount

Syntax

RETCODE **SQLRowCount***(hstmt,pcrow)*

Description

SQLRowCount returns the number of rows affected by an Update, Insert, or Delete statement or by a SQL_UPDATE, SQL_ADD, or SQL_DELETE operation in SQLSetPos. If SQLRowCount is called during a fetch cycle, the value returned is the number of rows returned to the application at the current position.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.

Argument	Type	Use	Description
pcrow	SDWORD FAR *	Input	Pointer to storage of the row counter.

SQLNumParams

Syntax

RETCODE **SQLNumParams***(hstmt,pccol)*

Description

SQLNumParams returns the number of parameters in a SQL statement.

Parameters

This table describes the parameters:

Argument	Type	Use	Description
hstmt	HSTMT	Input	Statement handle
pccol	SWORD FAR*	Output	Number of parameters in the statement.

SQLNumResultCols

Syntax

RETCODE **SQLNumResultCols***(hstmt,pccol)*

Description

SQLNumResultCols returns the number of columns in the result set. SQLNumResultCols can be called successfully only when the statement handle is in the prepared or executed state. An application typically would use the value returned in pccol in a loop and call SQLDescribeCol for each column in the result set.

An application retrieves an entire row of values using a technique called binding. Binding associates the data from the data source with variables in the application program. There are two directions of binding: input and output. Input data must always be bound. On output, when a result column is bound, the variable receives the value for that column each time a new row is fetched. The following example shows how this technique differs from SQLGetData:

```
/* for all columns in the current result set */
for (i = 0; i < columns; i++)
   SQLBindCol(hstmt, ...,&value[i], ...)
while (SQL_SUCCESS == (rc = SQLFetch(hstmt)))
/* value[ i .. n] contains data for current row */
```

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle
pccol	SWORD FAR*	Output	Number of columns in the result set.

SQLBindCol

Syntax

RETCODE **SQLBindCol***(hstmt,icol,fCType,rbgValue,cbValueMax,pcbValue)*

Description

SQLBindCol assigns the storage and data type for a column in a result set.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
icol	UWORD	Input	Column number of result data.

Argument	***Type***	***Use***	***Description***
fCType	SWORD	Input	The C data type of the result data.
rbgValue	PTR	Both	A pointer to storage for the result column.
cbValueMax	SDWORD	Input	Maximum length of the rgbValue buffer.
pcbValue	SDWORD FAR*	Both	A pointer to a buffer for the SQL_NULL_DATA or the number of bytes available to return in rgbValue prior to calling SQLFetch.

SQLBindParameter

Syntax

`RETCODE` **`SQLBindParameter`***`(hstmt,ipar,fParamType,fCType,fSqlType,cbColDef,ibScale, rbgValue,cbValueMax,pcbValue)`*

Description

An application calls SQLBindParameter to bind each parameter marker in a SQL statement. Bindings remain in effect until the application calls SQLBindParameter again or until the application calls SQLFreeStmt with the SQL_DROP or SQL_RESET_PARAMS option.

An application can use SQLBindParameter to supply the prompt values for a PeopleSoft query. SQLBindParameter calls the new function, ODBCBindParm. ODBCBindParm converts the ODBC C data type, fCType, to the ODBC SQL data type, fSqlType. It then maps the ODBC SQL data type to a supported PeopleSoft RDM data type and calls the appropriate internal bind function.

An ODBC driver is required to support conversions from all ODBC C data types to the ODBC SQL data types that they support.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle
ipar	UWORD	Input	Parameter number, ordered sequentially left to right, starting at 1.
fParamType	SWORD	Input	The type of the parameter.
fCType	SWORD	Input	The C data type of the parameter.
fSqlType	SWORD	Input	The SQL data type of the parameter.
cbColDef	UDWORD	Input	The precision of the column or expression of the corresponding parameter marker.
ibScale	SWORD	Input	The scale of the column or expression of the corresponding parameter marker
rbgValue	PTR	Both	A pointer to a buffer for the parameter's data.
cbValueMax	SDWORD	Input	Maximum length of the rgbValue buffer.
pcbValue	SDWORD FAR*	Both	A pointer to a buffer for the parameter's length.

An application may also supply prompt values as literal strings embedded in the SQL statement string. For example:

```
SQLExecDirect(hstmt, "{call  query.myquery(8001, NEWGN)}", SQL_NTS);
```

If prompt values are not provided, PeopleSoft Query prompts the user for each required value at the time of statement execution.

See Also

Chapter 6, "Using the Open Query ODBC Driver and API," ODBC to RDM Data Types, page 114

Execution Models

ODBC supports three execution models. Each accomplishes the same tasks but differs from the others with regard to when and where (on the client or on the server) each step is performed.

ExecDirect

In this model, the SQL statement is specified, sent to the server, and executed all in one step. This model is best suited for SQL statements for a particular purpose or SQL statements that are executed only once. Parameters can be used, but they act merely as placeholders that the driver replaces with the parameter values before it sends the SQL statement to the server.

The DBMS discards the optimization information that is used to execute the SQL statement after execution is complete. If the same statement is specified again with SQLExecDirect, the entire process of parsing and optimizing happens again.

Prepare/Execute

In this model, the SQL statement is prepared (sent to the server, parsed, and optimized) first and executed later. When the statement is executed, what flows to the server is not the SQL statement itself, but a way of referencing the statement so that the access plan can be executed immediately. Parameters are often used in these SQL statements, so the only items that flow to the server are the reference to the access plan and the parameter values, not the entire SQL statement.

The Prepare/Execute model should be used when repeated execution of the same SQL statement is needed and when the SQL statement must be constructed dynamically during runtime. To use this model, the application calls SQLPrepare and then (presumably in a loop) calls SQLExecute.

Stored Procedures

The stored procedure model is like the Prepare/Execute model except that with stored procedures, the preparation step can be done independently from the application and the stored procedure persists beyond the runtime of the application. To use stored procedures in ODBC, the application calls SQLExecDirect but uses the SQL statement to specify the stored procedure name, as illustrated in the following example:

```
SQLExecDirect(hstmt, "{call query.proc1(?,?,?)}", SQL_NTS);
```

Using Retrieval Procedures

For row-returning statements, such as Select statements or stored procedures, ODBC provides three ways to retrieve data. Using a single function call, an application can retrieve a single value, an entire row of values, or multiple rows of values. The PeopleSoft driver supports only the first two methods: single value and entire row.

One way that an application can retrieve data is by using a function call (SQLGetData) for every column in every row. The application supplies function arguments that specify the column number and a variable in which to receive the data. After the function call has been successfully executed, the value for the given column is returned in the variable. The application uses two loops to retrieve an entire result set, as in this example:

```
/* For all rows */
while (SQL_SUCCESS == (rc = SQLFetch(hstmt)))
   /* for all columns in current results set */
   for (colnum = 1; colnum <= columns; colnum++)
      SQLGetData(hstmt, colnum, ..., &value, ..)
```

SQLGetData is also used for the piecemeal retrieval of large text and binary data (such as images). It is often difficult or impossible for an application to allocate a single piece of memory big enough to hold a large data object, such as a 50-page document or a high-density bitmap.

Retrieval Procedures

This section discusses the retrieval procedures in alphabetical order.

SQLFetch

Syntax

`RETCODE` **`SQLFetch`**_`(hstmt)`_

Description

SQLFetch fetches a row of data from a result set. The driver returns data for all columns that were bound to storage locations with SQLBindCol. SQLFetch positions the cursor on the next row of the result set. When the cursor is positioned to the last row of the result set, SQLFetch returns SQL_NO_DATA_FOUND.

If the application called SQLBindCol to bind columns, SQLFetch stores data in the locations specified by the calls to SQLBindCol. If the application does not call SQLBindCol to bind any columns, SQLFetch does not return any data; it just moves the cursor to the next row. An application can call SQLGetData to retrieve data not previously bound to a storage location.

Parameters

This table describes the parameter:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle

SQLGetData

Syntax

```
RETCODE SQLGetData(hstmt,icol,fCType,rbgValue,cbValueMax,pcbValue)
```

Description

SQLGetData returns result data for a single unbound column in the current row. The application must call SQLFetch to position the cursor on a row of data before it calls SQLGetData. You can use SQLBindCol for some columns and use SQLGetData for others within the same row. This function can be used to retrieve character or binary data values in parts from a column with a character, binary, or data-source-specific data type (for example, data from SQL_LONGVARBINARY or SQL_LONGVARCHAR columns).

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
hstmt	HSTMT	Input	Statement handle.
icol	UWORD	Input	Column number of result data.
fCType	SWORD	Input	The C data type of the result data.
rbgValue	PTR	Both	A pointer to storage for the result column.
cbValueMax	SDWORD	Input	Maximum length of the *rgbValue* buffer.
pcbValue	SDWORD FAR*	Both	A pointer to a buffer for the SQL_NULL_DATA or the number of bytes available to return in *rgbValue* prior to calling SQLFetch.

Status and Error Retrieval Procedures

When any ODBC call fails, the driver retains error information until the next ODBC call. The error state and error text is retrieved from the driver with the SQLError function.

SQLError

Syntax

```
RETCODE SQLError(henv,hdbc,hstmt,szSqlState,pfNativeError,szErrorMsg,
cbErrorMsgMax,pcbErrorMsg)
```

Description

SQLError returns error or status information. An application typically calls SQLError when a previous call to an ODBC function returns SQL_ERROR or SQL_SUCCESS_WITH_INFO. The application can, however, call SQLError after any ODBC function call.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
henv	HENV	Input	Environment handle or SQL_NULL_HENV.
hdbc	HDBC	Input	Connection handle or SQL_NULL_HDBC.
hstmt	HSTMT	Input	Statement handle or SQL_NULL_HSTMT.
szSqlState	UCHAR FAR *	Output	SQLSTATE as null terminated string.
pfNativeError	SDWORD FAR *	Output	Native error code (specific to the data source).
szErrorMsg	UCHAR FAR *	Output	Pointer to storage for the error message text.

Argument	***Type***	***Use***	***Description***
cbErrorMsgMax	SWORD	Input	Maximum length of the *szErrorMsg* buffer. This must be less than or equal to SQL_MAX_MESSAGE_LENGTH - 1.
pcbErrorMsg	SWORD FAR *	Output	Pointer to the total number of bytes (excluding the null termination byte) available to return in *szErrorMsg*. If the number of bytes available to return is greater than or equal to *cbErrorMsgMax*, the error message text in *szErrorMsg* is truncated to *cbErrorMsgMax* – 1 bytes.

Transaction and Connection Termination Procedures

Each query object that runs in ODBC creates a transaction. To ensure that all memory that is associated with a transaction is freed and locks are released, an application should call SQLTransact.

SQLTransact

Syntax

RETCODE **SQLTransact***(henv,hdbc,fType)*

Description

SQLTransact requests a commit or rollback operation for all active operations on all statement handles that are associated with a connection. SQLTransact can also request that a commit or rollback operation be performed for all connections that are associated with the environment handle. In the case of query objects, the transaction is automatically closed upon termination of the statement handle.

Parameters

This table describes the parameters:

Argument	***Type***	***Use***	***Description***
henv	HENV	Input	Environment handle or SQL_NULL_HENV.
hdbc	HDBC	Input	Connection handle or SQL_NULL_HDBC.
fType	UWORD	Input	Flag for SQL_COMMIT or SQL_ROLLBACK.

SQLDisconnect

Syntax

```
RETCODE SQLDisconnect(hdbc)
```

Description

SQLDisconnect closes the connection that is associated with a specific connection handle.

If an application calls SQLDisconnect before it has freed all statement handles that are associated with the connection, the driver frees those statement handles after it successfully disconnects from the data source. However, if one or more of the statement handles that are associated with the connection are still executing asynchronously, SQLDisconnect will return SQL_ERROR.

Parameters

This table describes the parameter:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.

SQLFreeConnect

Syntax

```
RETCODE SQLFreeConnect(hdbc)
```

Description

SQLFreeConnect releases a connection handle and frees all memory that is associated with the handle. This is called after SQLDisconnect.

Parameters

This table describes the parameter:

Argument	***Type***	***Use***	***Description***
hdbc	HDBC	Input	Connection handle.

SQLFreeEnv

Syntax

```
RETCODE SQLFreeEnv(henv)
```

Description

SQLFreeEnv frees the environment handle and releases all memory that is associated with the environment handle.

Parameters

This table describes the parameter:

Argument	***Type***	***Use***	***Description***
henv	HENV	Input	Environment handle.

ODBC Compliance

The ODBC API defines a set of core functions that correspond to the functions in the X/Open and SQL Access Group Call Level Interface specification. ODBC also defines two extended sets of functionality, Level 1 and Level 2.

For the specific ODBC API descriptions and implementation details, refer to the Microsoft Open Database Connectivity Software Development Kit.

The following lists summarize the functionality that is included in each conformance level.

Core API functions allow the following:

- Allocation and releasing of environment, connection, and statement handles.
- Conversion to data sources. Use of multiple statements on a connection.
- Preparation and immediate execution of SQL statements.
- Assignment of storage for parameters in a SQL statement and result columns.
- Retrieval of data from a result set. Retrieval of information about a result set.
- Commit or rollback transactions.
- Retrieval of error information.

The Level 1 API allows all the core functions, plus the following:

- Connection to data sources with driver-specific dialog boxes.
- Set and inquire values of statement and connection options.
- Transmission of part or all of a parameter value (useful for long data).
- Retrieval of part or all of a result column value (useful for long data).
- Retrieval of catalog information (columns, special columns, statistics, and tables).
- Retrieval information about driver and data source capabilities, such as supported data types, scalar functions, and ODBC functions.
- The Level 2 API allows all the core and Level 1 functions, plus the following:
- Ability to browse connection information and list available data sources.
- Transmission of arrays of parameter values. Retrieval of arrays of result columns values.
- Retrieval of the number of parameters and description of individual parameters.
- Scrollable cursor.
- Retrieval of the native form of a SQL statement.
- Retrieval of catalog information (privileges, keys, and procedures).
- Translation DLL calls.

To claim that it conforms to a given API or SQL conformance level, a driver must support all the functionality in that conformance level, regardless of whether that functionality is supported by the DBMS that is associated with the driver. However, conformance levels do not restrict drivers to the functionality in the levels to which they conform. Drivers support as much functionality as they can; applications can determine the functionality that is supported by a driver by calling SQLGetInfo, SQLGetFunctions, and SQLGetTypeInfo.

ODBC to RDM Data Types

The following table shows the mapping from ODBC C data types to ODBC SQL and PeopleSoft RDM data types:

RDM Type	***fSqlType***	***Type***
RDM_CHAR	SQL_CHAR	Unsigned char FAR*
RDM_LONG_CHAR	SQL_VARCHAR	Unsigned char FAR*
RDM_NUMBER, RDM_SIGNED_NUMBER	SQL_NUMERIC	Unsigned char FAR*
RDM_DATE	SQL_DATE	Struct tag DATE_STRUCT { UWORD year; UWORD month; UWORD day; }
RDM_TIME	SQL_TIME	Struct TIME_STRUCT { UWORD hour; UWORD minute; UWORD second; }
RDM_DATETIME	SQL_TIMESTAMP	Struct TIMESTAMP_STRUCT { SWORD year; UWORD month; UWORD day; UWORD hour; UWORD minute; UWORD second; UWORD fraction; }
RDM_IMAGE	SQL_LONGVARBINARY	Unsigned char FAR *

PeopleSoft Open Query ODBC API Example

The following example shows the ODBC API calls needed to execute a PeopleSoft query using the PeopleSoft Open Query ODBC driver. The following query requires two bind variables: business unit and department ID. The sample query returns an answer set of three columns: employee ID, name, and monthly rate.

```
/**********************************************************************
* Function:     OpenQuerySample                                       *
*                                                                     *
* Description:  Sample program illustrating the usage of PeopleSoft  *
*               Open Query ODBC API.                                  *
*               Sample code uses basic PeopleSoft Query ODBC interface *
*               functions.  Most error checking is excluded to make   *
*               code easier to follow; in a typical application,      *
*               every return code would be checked.                   *
*                                                                     *
* Returns:      TRUE if successful                                    *
**********************************************************************/


BOOL WINAPI     OpenQuerySample(HWND hWnd)
{
HENV            hEnv;                   // Environment handle for application
HDBC            hDbc;                   // Connection handle
HSTMT           hStmt;                  // Statement handle
RETCODE         retcode;                // Return code
char            szConnectString[] =
"DSN=PeopleSoft PeopleTools;DBTYPE=ORACLE;DBNAME=PTDMO7;UID=PTDMO;PWD=PTDMO;";
char            szConnStringOut[256];   // completed connection string
SWORD           nConnStringLen;         // length of completed connect string


// Allocate environment, database connection
retcode = SQLAllocEnv(&hEnv);
if ((retcode = SQLAllocConnect(hEnv, &hDbc)) != SQL_SUCCESS)
    // error--this would normally abort program with message
    return(FALSE);

// Connect to the database
retcode = SQLDriverConnect(hDbc, hWnd, szConnectString,
        strlen(szConnectString), szConnStringOut, sizeof(szConnStringOut),
        &nConnStringLen, SQL_DRIVER_COMPLETE);

retcode = SQLAllocStmt(hDbc, &hStmt);

ProcessQuery(hStmt);

// Close the connection, release resources
retcode = SQLFreeStmt(hStmt);
retcode = SQLDisconnect(hDbc);
retcode = SQLFreeConnect(hDbc);
retcode = SQLFreeEnv(hEnv);

return(TRUE);
}


/**********************************************************************
* Function:     ProcessQuery                                          *
*                                                                     *
* Description:  Run a query and retrieve answer set.                  *
*                                                                     *
* Returns:      TRUE if successful, else FALSE                        *
**********************************************************************/

BOOL            ProcessQuery(HSTMT hStmt)
{
RETCODE         retcode;                // Return code
char            szSelect[] = "{call query.myquery(?,?)";
// binding of input variables must occur before statement execution
for (i = 0; i < 2; i++)
    retcode = SQLBindCol(hStmt, i, datatype, &value, sizeof(value), &valuelen);

retcode = SQLExecDirect(hStmt, szSelect, strlen(szSelect));

while (retcode = SQLFetch(hStmt) == SQL_SUCCESS)
    // process data for a fetched row....

return(retcode == SQL_NO_DATA_FOUND);
}
```

Index