PeopleTools 8.4: PeopleSoft Integration Tools and Utilities

PeopleSoft®

PeopleTools 8.4: PeopleSoft Integration Tools and Utilities

SKU Tr84INT-B 0302

PeopleBooks Contributors: Teams from PeopleSoft Product Documentation and Development.

Contents

PeopleSoft Integration Tools and Utilities Preface

In addition to PeopleSoft Integration Broker, Component Interfaces and Business Interlinks, PeopleTools includes several supplemental integration tools and utilities. PeopleSoft Integration Tools and Utilities describes how you can use these products to facilitate the implementation of your integration strategy.

The "About This PeopleBook" section contains general product line information, such as related documentation, common page elements, and typographical conventions. This book also contains a glossary with useful terms that are used in PeopleBooks.

See **PeopleSoft Glossary**.

About This PeopleBook

This book provides you with the information that you need for implementing and using *PeopleTools 8.4* applications. Complete documentation for this release is provided on the CD-ROM PT84PBR0.

Note. Your access to PeopleSoft PeopleBooks depends on which PeopleSoft applications you've licensed. You may not have access to all of the PeopleBooks.

This section contains information that you should know before you begin working with PeopleSoft products and documentation, including PeopleSoft-specific documentation conventions, information specific to each PeopleSoft product line, and information on ordering additional copies of our documentation.

Before You Begin

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

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

Because we assume that you already know how to navigate the PeopleSoft system, much of the information in these books is not procedural. That is, these books do not typically provide step-by-step instructions on using tables, pages, and menus. Instead, we provide you with the information that you need to use the system most effectively and to implement your

PeopleSoft application according to your organizational or departmental needs. PeopleBooks expand on the material covered in PeopleSoft training classes.

PeopleSoft Application Fundamentals

Each PeopleSoft application PeopleBook provides implementation and processing information for your PeopleSoft database. However, there is additional, essential information describing the setup and design of your database contained in a companion volume of documentation called *PeopleSoft Application Fundamentals.*

PeopleSoft Application Fundamentals contains important topics that apply to many or all PeopleSoft applications across each product line. Whether you are implementing only one PeopleSoft application, some combination of products within a product line, or an entire PeopleSoft system, you should be familiar with the contents of this central PeopleBook. It contains fundamental information such as setting up control tables and administering security.

The PeopleSoft Applications Fundamentals PeopleBook contains common information pertinent to all applications in each product line, such as defining general options. If you're upgrading from a previous PeopleSoft release, you may notice that we've removed some topics or topic headings from the individual application PeopleBooks and consolidated them in this single reference book. You'll now find only application-specific information in your individual application PeopleBooks. This makes the documentation as a whole less redundant. Throughout each PeopleBook, we provide cross-references to *PeopleSoft Application Fundamentals* and other PeopleBooks.

Related Documentation

You can order printed, bound versions of the complete PeopleSoft documentation delivered on your PeopleBooks CD-ROM and additional copies of the PeopleBooks CDs through the Documentation section of the PeopleSoft Customer Connection website: http://www.peoplesoft.com/corp/en/login.asp

You can find updates and additional documentation for this release, as well as previous releases, on PeopleSoft Customer Connection (http://www.peoplesoft.com/corp/en/login.asp). Through the Documentation section of Customer Connection, you can download files to add to your PeopleBook library. You'll find a variety of useful and timely materials, including updates to the full PeopleSoft documentation delivered on your PeopleBooks CD.

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

Hard-copy Documentation

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

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

Internet	From the main PeopleSoft Internet site, go to the Documentation section of Customer Connection. You can find order information under the Ordering PeopleBooks topic. Use a Customer Connection ID, credit card, or purchase order to place your order. PeopleSoft Internet site: http://www.peoplesoft.com/.
Telephone	Contact Consolidated Publishing Incorporated (CPI) at **800 888 3559.**
Email	Send email to CPI at **callcenter@conpub.com**.

PeopleBooks Standard Field Definitions

Throughout our product documentation, you will encounter fields and buttons that are used on many application pages or panels. This section lists the most common fields and buttons and provides standard definitions.

Field	***Definition***
As of Date	The last date for which a report or process includes data.
Business Unit	An identification code that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization.
Description	Freeflow text up to 30 characters.
Effective Date	Date on which a table row becomes effective; the date that an action begins. For example, if you want to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when you can view and change the information. Pages or panels and batch processes that use the information use the current row. For more information about effective dates, see **Understanding Effective Dates in *Using PeopleSoft Applications.***

Field	***Definition***
EmplID (employee ID)	Unique identification code for an individual associated with your organization.
Language or **Language Code**	The language in which you want the field labels and report headings of your reports to print. The field values appear as you enter them. Language also refers to the language spoken by an employee, applicant, or non-employee.
Process Frequency group box	Designates the appropriate frequency in the **Process Frequency** group box: **Once** executes the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to **Don't Run**. **Always** executes the request every time the batch process runs. **Don't Run** ignores the request when the batch process runs.
Report ID	The report identifier.
Report Manager	This button takes you to the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list).
Process Monitor	This button takes you to the Process List page, where you can view the status of submitted process requests.
Run	This button takes you to the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format.
	For more information about the Report List page, the Process List page, and the Process Scheduler, see Process Scheduler Basics in the PeopleTools documentation.
Request ID	A request identification that represents a set of selection criteria for a report or process.
User ID	The system identifier for the individual who generates a transaction.
SetID	An identification code that represents a set of control table information or TableSets. A TableSet is a group of tables (records) necessary to define your company's structure and processing options.
Short Description	Freeflow text up to 15 characters.

Typographical Conventions and Visual Cues

We use a number of standard conventions and visual cues in our online documentation.

The following list contains our typographical conventions and visual cues:

`(monospace font)`	Indicates a PeopleCode program or other program example.
Bold	Indicates field names and other page elements, such as buttons and group box labels, when these elements are documented below the page on which they appear. When we refer to these elements elsewhere in the documentation, we set them in Normal style (not in bold). We also use boldface when we refer to navigational paths, menu names, or process actions (such as **Save** and **Run**).
Italics	Indicates a PeopleSoft or other book-length publication. We also use italics for *emphasis* and to indicate specific field values. When we cite a field value under the page on which it appears, we use this style: ***field value.*** We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number *0,* not the letter *O.*
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press W.
Cross-references	The phrase **For more information** indicates where you can find additional documentation on the topic at hand. We include the navigational path to the referenced topic, separated by colons (:). Capitalized titles in *italics* indicate the title of a PeopleBook; capitalized titles in normal font refer to sections and specific topics within the PeopleBook. Here's an example: **For more information,** see **Documentation on CD-ROM** in *About These PeopleBooks:* Additional Resources.

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

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

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

Page and Panel Introductory Table

In the documentation, each page or panel description in the application includes an introductory table with pertinent information about the page. Not all of the information will be available for all pages or panels.

Usage	Describes how you would use the page or process.
Object Name	Gives the system name of the panel or process as specified in the PeopleTools Application Designer. For example, the Object Name of the Detail Calendar panel is DETAIL_CALENDAR1.
Navigation	Provides the path for accessing the page or process.
Prerequisites	Specifies which objects must have been defined before you use the page or process.
Access Requirements	Specifies the keys and other information necessary to access the page. For example, **SetID** and **Calendar ID** are required to open the Detail Calendar page.

Comments and Suggestions

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

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

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

DOC@PEOPLESOFT.COM

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

CHAPTER 1

File Layouts and Data Interchange

This chapter provides an overview of file layouts and data interchange and describes how to:

- Create a file layout definition.
- Customize a file layout.
- Perform data interchange.
- Create a sample layout.

Understanding File Layouts

A *file layout* is a definition that represents the data structure of a flat (text) file to be processed. When reading from or writing to a flat file, your PeopleCode can use a file layout as a template to identify and correctly process the file's records and fields. File layouts work with hierarchical and non-hierarchical data, and can handle files that combine data records with non-data (audit or control) records.

To access data in a file, you don't have to create a file layout. PeopleTools supports reading and writing to plain text files, as well as to files that have a format based on a file layout.

- If the file is a plain text file, data is read or written using text strings.
- If the file is based on a file layout, you can use text strings, rowset or record objects.

Using a file layout greatly simplifies reading, writing and manipulating hierarchical transaction data with PeopleCode. It can facilitate transactions between your PeopleSoft application and a third party system when the third party system doesn't support Integration Broker or component interfaces.

Applying File Layouts to Data Interchange

In addition to manipulating transaction data, you can employ file layouts to move data between your PeopleSoft database and external systems (data interchange), using flat files as the transmission medium. File layouts enable you to:

- Export hierarchical PeopleSoft data to several flat file formats.
- Map incoming hierarchical file data directly to PeopleSoft tables.
- Preview and troubleshoot the input data and its formatting before importing it.

- Automatically generate the Application Engine and PeopleCode programs needed to perform data import.
- Use batch processes to perform large volume data import and export.

See Also

Performing Data Interchange

PeopleTools PeopleBook: PeopleCode Reference, ”File Class”

Identifying Fields in a Flat File

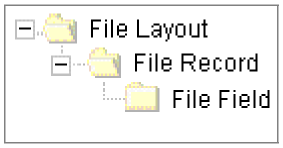
A flat file, in the simplest sense, is a collection of fields in text format. The file must be formatted in a way that enables your PeopleCode to locate each field. PeopleSoft file layouts support three formats:

- *Fixed Position (FIXED):* Each field has a starting position and a length which together specify its location in the file. This is the default format for new file layouts.
- *Comma Separated Values (CSV):* Fields are located in the file by their sequence, separated from each other by commas.
- *XML Tagged (XML):* A field is located not by its position or sequence within a record, but by the named XML tags surrounding it.

To preserve relationships between fields, we need a way to logically group fields into a collection. In relational databases, these collections are records. Each line within a file can be considered a collection of fields.

Note. With some file formats, the logical concept of a record may actually span multiple physical lines, but the concept of collections of fields remains.

The conceptual structure of all three file formats is represented in a file layout as follows:



File layout structure

A file layout is a collection of file records, which in turn are a collection of fields, each of which has a describable location.

Note. To avoid confusion with the standard terms record and field, when working with a file layout we refer to a collection of fields as a file record and to fields as file fields.

Fixed Format Positional File (FIXED)

This is the most common type of flat file currently processed by PeopleSoft EDI Manager. Almost all EDI type processing uses this file type where each data element is oriented by a fixed, or column dependent, position within the file.

Fixed Format Attributes

Attribute	*Description*	*EDI Manager Equivalent*
File Layout Format	File format (FIXED).	None.
File Record ID	A group of numbers that can be used to identify the file record.	RowID.
ID Start Position	The column starting position of the file record ID.	Treated as a field within each map.
(Record) ID Length	The length of the file record ID.	Treated as a field within each map.
File Record Name	A user specified name for the file record.	PeopleSoft record name.
File Field	A user specified name for the file field.	PeopleSoft record's field name.
(Field) Start Position	The column starting position of the file field.	Starting position.
Field Length	The length of the file field.	Length of field.
Field Format	The formatting options for both inbound and outbound field processing.	Based on field type.

Considerations for FIXED Format

You should be aware of the following when working with files of FIXED format.

- Be careful when you change the length or starting position of any file fields, or if you insert a new file field between two existing ones. It's possible to overlay fields. Results are unpredictable.
- When you insert a record into a file layout, fields of type Long are converted to type Character, with a length of 0. You must set a field length greater than 0 before you can save the file layout.

Variable Format Delimited File (CSV)

In this type of file, each data element is surrounded with a separator, a delimiter or both. File Record IDs can be used to determine which table data is moved to or from, however, in most cases this type of file contains homogenous records.

In the following example, the qualifier is double quotes (“) and the delimiter is a comma (,).

```
"NAME","ADDRESS","PHONE"
```

File layout definitions store the file record ID (when used) and the relative sequence number of the field. (In the text example above, “PHONE” is sequence number 3).

CSV Format Attributes

Attribute	*Description*	*EDI Manager Equivalent*
File Format	File format (CSV).	None.
File Record ID	A group of numbers that can be used to identify the file record.	RowID.
ID Sequence Number	The sequence number of the field that contains the file record ID.	Treated as a field within each map.
Qualifier	The character that surrounds a field to mark its extent. This can be set at the file layout, file record or file field level.	Delimiter.
Delimiter	The character used to separate fields from each other. This can be set at the file layout or file record level.	Separator.
File Record Name	A user specified name for the file record.	None
File Record Field	A user specified name for the file field.	None
Field Format	The formatting options for both inbound and outbound field processing.	Based on Field type

Considerations for CSV Format

You should be aware of the following when working with files of CSV format:

- Both the qualifier and the delimiter accept a blank as a valid value.
- If a field is NULL, you don't have to use qualifiers. In the following example, Field2 is NULL.

```
Field1,,Field3,Field4. . .
```

Tagged Hierarchical Data File (XML)

This type of file contains data represented in a hierarchical or tree type structure. A tag surrounds each data element. A file record tag might group multiple entries.

File layout definitions tie the identifier along with parent and child relationships to the file record and file field.

There is no PeopleSoft EDI Manager equivalent for this format.

XML Format Attributes

Attribute	*Description*
File Format	The file format (XML).
File Record ID	The tag name representing the file record.
Field Identifier	The tag name representing the file field.
Field Format	The formatting options for both inbound and outbound field processing.
File Record Name	A user specified name for the file record.
Field Name	A user specified name for the file field.

Considerations for XML Format

You should be aware of the following when working with files in XML format:

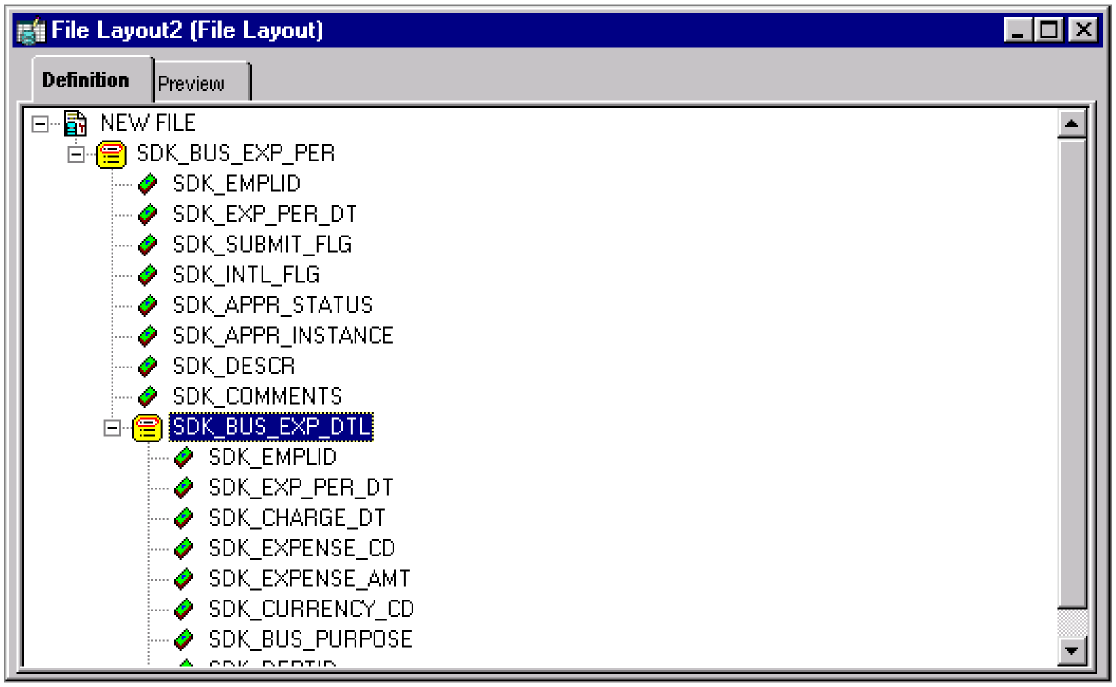
- Your XML input file must contain *at least* the fields that are specified in the file layout definition you're using. If the file is missing any fields, the input rowset won't contain any data.
- If your XML input file contains extra fields that aren't specified in the file layout definition, the ReadRowset method will ignore the extra fields.
- When you insert a record into a file layout, fields of type Long are converted to type Character, with a length of 0. You must set a field length greater than 0 before you can save the file layout.

Constructing File Layouts

This section describes how to:

- Create a new file layout.
- Add file records and file fields to the layout.
- Reorder the file layout nodes.
- Name file layouts, records, and fields.
- Understand certain field formatting issues.

Creating a New File Layout Definition



```
NEW FILE
  SDK_BUS_EXP_PER
    SDK_EMPLID
    SDK_EXP_PER_DT
    SDK_SUBMIT_FLG
    SDK_INTL_FLG
    SDK_APPR_STATUS
    SDK_APPR_INSTANCE
    SDK_DESCR
    SDK_COMMENTS
    SDK_BUS_EXP_DTL
      SDK_EMPLID
      SDK_EXP_PER_DT
      SDK_CHARGE_DT
      SDK_EXPENSE_CD
      SDK_EXPENSE_AMT
      SDK_CURRENCY_CD
      SDK_BUS_PURPOSE
```

File layout with two records in hierarchy

To create a new file layout:

1. In Application Designer, select File, New, File Layout.

2. Click OK to create the new file layout.

   The File Layout definition window appears, with the Definition tab selected. A new file layout contains only one element — the root node, to which file records are attached. The default root node name is “NEW FILE.”

3. Save the file layout definition.

   You’ll be prompted to give the file layout a name, which will also be the name of the root node.

Note. The default file layout format is FIXED. See Specifying File Layout Properties for details.

Adding File Records and File Fields

You can add file records and file fields to your file layout in two ways:

- You can base them on existing record and field definitions in your database.

- You can construct them directly in your file layout, without reference to any database records or fields. A file record you construct this way is added as a *segment,* but the result is an ordinary file record.

You can use both methods in any combination when creating file records and file fields in a file layout definition. Each file record and file field is generically referred to as a *node.*

Important! Regardless of the method you use, file records and file fields exist only as definitions within a file layout, and have no connection with any database record or field. Even with file records based on records in your database, a change to the database record definition will *not* be reflected in the file layout.

When to Use a Segment Instead of a Record

Suppose that in the file provided to you, some of the file records contain new data and need to be inserted, while others contain data that updates existing data. You could add a segment with a single field (like AUDIT_ACTION) that indicates whether the file record is new or changed. When you process the file, you can use PeopleCode to look at this field and, based on its value, do the appropriate action.

Another example: suppose you wanted to include two fields from the PERSONAL_DATA table in your file, but not all the other fields. You have two choices: insert the PERSONAL_DATA table and manually delete all the unwanted fields, or insert a segment, name it PERSONAL_DATA, then insert the two fields you want.

Segments in Data Interchange

If you're creating a file layout for the purpose of data interchange, you can use segments, but each file record must correspond to a record with the same name in your PeopleSoft database, and its file fields must have the same names and data formats as the database record's fields. The file record may contain a subset of the fields in the corresponding database record. It may also contain a subset or a superset of the fields provided in the corresponding file data.

Adding File Records

To add a file record to your file layout definition, use one of the following methods:

- Drag and drop a database record — drag a record definition from the Project window into the file layout's Definition window, dropping it on the root node or any existing file record or file field. The new file record appears at the same level as the node you dropped it on, following all the other file records at that level. All of the record's constituent fields are inserted as well.

- Insert a database record — with the root node or a file record highlighted, select Insert, Record. When you select a record, the new file record appears following the highlighted file record, at the same level. All of the record's constituent fields are automatically inserted as well.

- Insert a segment — with the root node or a file record highlighted, select Insert, Segment, and enter a file record name. When you click OK, the new file record appears following the highlighted file record, at the same level.
- Insert a child segment — with a file record highlighted, select Insert, ChildSegment, and enter a file record name. When you click OK, the new file record appears one level below the highlighted file record, before any others at that level.

Note. When you add a file record at the root level, it appears immediately below the root node, before all the other file records.

Adding File Fields

To add a file field to your file layout definition, use one of the following methods:

- Drag and drop a database field — drag a field definition from the Project window into the file layout's Definition window, dropping it on any existing file record or file field. Confirm the field name or enter a different one, and click OK. The new file field appears following the node you dropped it on.
- Insert a database field — with a file record or file field highlighted, select Insert, Database Field. Confirm the field name or enter a different one, and click OK. The new file field appears following the highlighted node.
- Insert a file field — with a file record or file field highlighted, select Insert, FileField, and enter a file field name. When you click OK, the new file field appears following the highlighted node.

Note. Each file field must have a unique name within its parent file record, but file fields in different file records can have the same name.

Reordering File Layout Nodes

The file layout definition provides a set of directional arrow buttons in the toolbar, which you can use to reposition any file record within the hierarchy of the file layout, or any file field within its parent file record.

The up and down arrows don't change the level of the selected item, just its order among other items at that level. The right and left arrows move the selected item lower and higher in the file layout hierarchy.

Note. When you reposition a file record in the file layout, its child records are also repositioned, and their child records, and so on.

Naming File Layouts, Records and Fields

File layout names can be 30 characters in length, and file record and file field names can be 15 characters in length, and all should follow PeopleSoft naming standards.

Each file record within a file layout must have a unique name, but one file record can have the same name as the file layout. Each file field within a given file record must have a unique name, but file fields in different file records can have the same name.

Using WriteRecord, ReadRowset and WriteRowset

If you use the WriteRecord, ReadRowset or WriteRowset file layout methods for writing to or reading from records, the application record and the file record *must have the same name*, and the application record fields and the file fields *must have the same names*. These methods only write to like-named records and like-named fields within a given record. If you rename a record or a field after you use it to create a file layout definition, you will have to rename your file record or file field to the exact same name.

In a file layout definition containing more than one record, records and fields that aren't like-named are ignored. Like-named records don't have to contain all the same fields, and like-named fields don't have to be the same length. PeopleSoft recommends that like-named fields be of the same type.

See Also

PeopleTools PeopleBook: Application Designer, "Creating Record Definitions," Naming Record Definitions

Understanding Field Formats

Numbers in FIXED Output Files

When you write numeric data to a FIXED format flat file, all numbers are written right justified in the file field. Numbers with decimal places specified will be written with zeros padding the unused decimal places.

For example, a sequence of records with numbers of varying precision are written this way:

```
001    53.2700BUY
002  2174.0933SELL
003   108.0000SELL
```

Date, Time, and Datetime Field Considerations

In accordance with ISO 8601 standards, the field lengths for Date, Time and Datetime fields are fixed in the file layout, regardless of file format:

- Date fields have a fixed length of 10.
- Time fields have a fixed length of 20.

- Datetime fields have a fixed length of 31.

Considerations for Using Dates With the ReadRowset Method

Single digits in dates in the form MMDDYY or MMDDYYYY must be padded with zeros. That is, if the date in your data is February 3, 2002, the form must be:

- 02/03/2002
- 02/03/02

The following is **not** valid: 2/3/02.

See Also

PeopleTools PeopleBook: PeopleCode Reference, "PeopleCode Classes," File Layout Error Processing

Customizing File Layouts

Each node in the file layout has an associated property dialog box. This section describes how to:

- Specify file layout properties.
- Specify file record properties.
- Specify file field properties.

Note. Some properties are only available for a specific file layout format. For example, a file definition tag is only available for a file with XML specified as the file layout format. When a property is only available for a particular format, that is noted in parentheses after the name of the property (such as File Definition Tag (XML)).

Specifying File Layout Properties

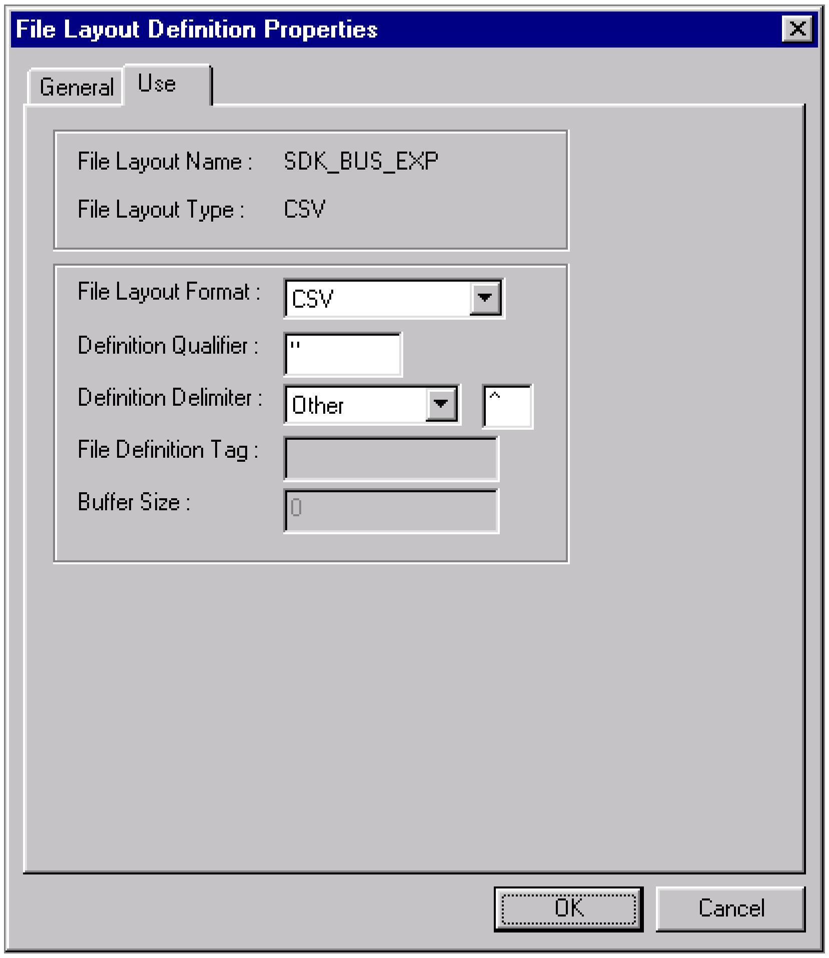
The File Layout Definition Properties dialog box contains all information stored at the file layout (root) level.

To access the dialog box:

- Use File, Object Properties.
- Press Alt+Enter
- Double-click the topmost (root) node of a file layout definition.
- Right-click an open file layout, then select Data Object Properties.

The General tab of the dialog box contains description information for the file layout.

The Use tab contains specific information for the file layout.



File Layout Name : SDK_BUS_EXP
File Layout Type : CSV
File Layout Format : CSV
Definition Qualifier : "
Definition Delimiter : Other ^
File Definition Tag :
Buffer Size : 0

File Layout Definition Properties: Use tab

File Layout Format	The type of file layout. Valid values are *FIXED*, *CSV*, or *XML*. See Understanding File Layouts for details about each format.
Definition Qualifier (CSV)	Select a qualifier to surround each field in the file record in this layout This value can be overridden at the file record and file field levels.
Definition Delimiter (CSV)	Select the default delimiter between each field in the file record in this layout. This value can be overridden at the file record level. The default value is *Comma*. If you specify *Other*, a blank field appears so you can enter a delimiter.
File Definition Tag (XML)	Enter the XML tag name associated with this layout (or transaction). This tag can be 30 characters in length. This tag must be unique in the file layout.

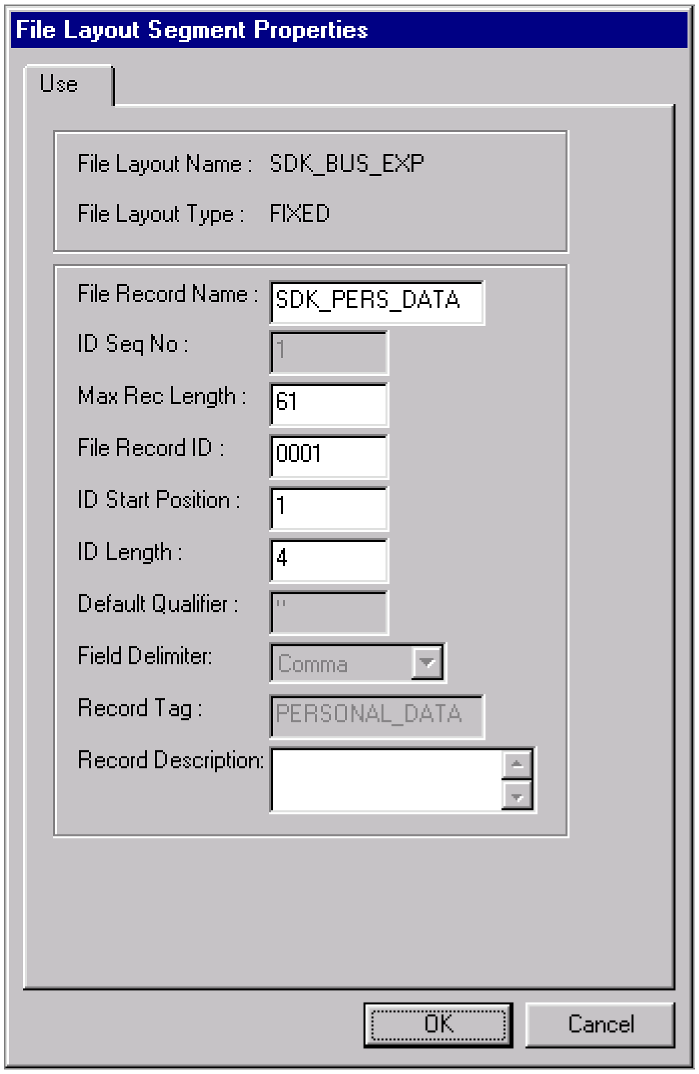
Buffer Size (XML)	Displays the size of the input buffer used at runtime. **Note.** This value shouldn't be edited directly.

Specifying File Record Properties

The File Layout Segment Properties dialog box contains information stored at the file record level.

To access the dialog box:

- Double-click the file record node.
- Select the file record node, right-click, then select Selected Node Properties.


File Layout Segment Properties

Use

File Layout Name : SDK_BUS_EXP

File Layout Type : FIXED

File Record Name :	SDK_PERS_DATA
ID Seq No :	1
Max Rec Length :	61
File Record ID :	0001
ID Start Position :	1
ID Length :	4
Default Qualifier :	"
Field Delimiter:	Comma
Record Tag :	PERSONAL_DATA
Record Description:	

File Layout Segment Properties dialog box

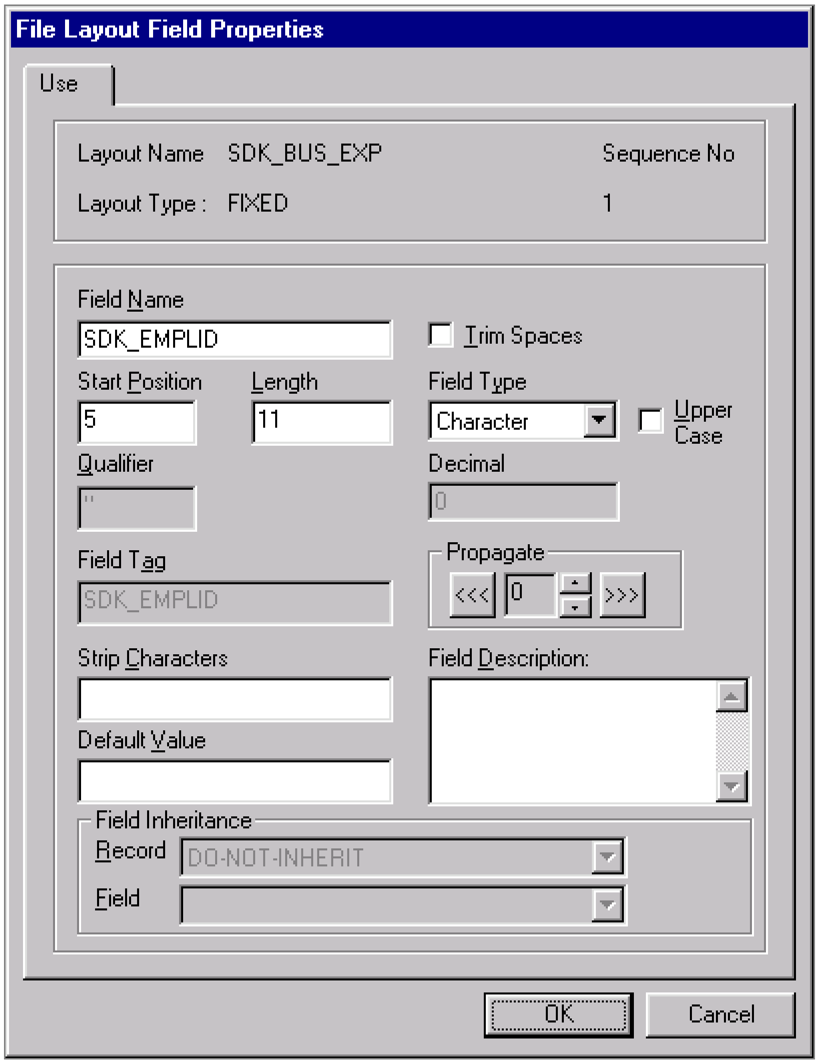
File Record Name	Enter a file record name associated with this file record. This name is used when accessing the file record from PeopleCode. Every file record in a file layout must have a unique name.
ID Seq No. (CSV)	Enter a sequence number for the field that contains the file record ID.
Max Rec Length	Displays the default maximum length of the combined field sizes of the record. This value is automatically updated. ***Warning!*** Any inbound or outbound data is truncated beyond this value.
File Record ID	Enter a number to uniquely identify the file record in the file layout. You can use this number in processing the file. This number is automatically written to the file if you use the WriteRecord or WriteRowset methods and the file type is FIXED or CSV.
ID Start Position (FIXED)	Enter the column or starting position in the file record where the file record ID starts.
ID Length (FIXED, CSV)	Displays the length of the file record ID. This number is automatically generated when you enter the File Record ID. **Note.** This value shouldn't be edited directly.
Default Qualifier (CSV)	Enter a qualifier used for the file record ID and the default for fields when no field qualifier is specified. This value overrides the definition qualifier specified in the File Layout Definition Properties dialog box. When you first create a file layout, this property is blank.
Field Delimiter (CSV)	Enter a delimiter used for all fields in the file record. This overwrites the definition delimiter specified on the File Layout Definition Properties dialog box.
Record Tag (XML)	Enter an XML tag name for this file record. The default value is the file record name. **Note.** Although each record name in a file layout must be unique, record tags do not have to be unique.
Record Description	Enter a description of the record, for documentation purposes only.

Specifying File Field Properties

The File Layout Field Properties dialog box contains information stored at the file field level.

To access the dialog box:

- Double-click the file field node.
- Select the file field node, right-click, then select Selected Node Properties.



Layout Name SDK_BUS_EXP
Layout Type : FIXED
Sequence No 1

Field Name: SDK_EMPLID
Trim Spaces: ☐
Start Position: 5
Length: 11
Field Type: Character
Upper Case: ☐
Qualifier: "
Decimal: 0
Field Tag: SDK_EMPLID
Propagate: 0
Strip Characters:
Field Description:
Default Value:
Field Inheritance
Record: DO-NOT-INHERIT
Field:

File Layout Field Properties dialog box

Most individual properties are usable by all field types. However, some are specific to a particular field type — for example, the **UpperCase** checkbox is only applicable for character fields, while the **Date Separator** field is only applicable for date fields, and so on. The dialog box pictured here shows the properties for a character type of field. However, the following description goes through all possible properties.

Field Name	Enter the name associated with this file field. This name is used when accessing the file field from PeopleCode. Every field within a file record must have a unique name; however, two different file records can contain the same file field.

UpperCase (Char)	Select to convert lowercase text to uppercase during inbound processing. Primarily used when customer data may be in lowercase, and PeopleSoft requires the data to be in uppercase.
Field Type	Select the data type of the file field.
Date Format (Date)	Select a date format, such as MMDDYY, DDYYMM, and so on.
Date Separator (Date)	Enter a character used to separate date values. The default value is /.
Decimal Position	Enter the number of decimal positions (to the right) of the decimal point. This property is only valid for fields defined as Number or Signed Number. **Note.** You're only allowed 31 characters plus a decimal point.
Field Length	Enter the maximum number of characters of this field. **Note.** You're only allowed 32 character precision for number and signed number fields, that is, a total of 32 characters both to the right and left of the decimal. Other fields, such as character fields, can be longer. **Note.** You can't set the field length for fields of type Date, Time, and Datetime. These field lengths are automatically set to the ISO standards for such fields.
Start Position (FIXED)	Enter the starting position (column) of the field within the file record. **Important!** If you specify a start position for a field that overwrites a previous field, no data is written to the file. Use **Propagate** to change the start positions for your file fields.
Propagate (FIXED)	If a field position or length is changed, enter an amount here to increment (positive number) or decrement (negative number) the current field and all fields before it (<<<) or after it (>>>).
Field Qualifier (CSV)	Enter the qualifier for the field, that is, the character that surrounds this field, separating it from other fields. Specifying this value overwrites the value specified in the file layout properties and file record properties.
Field Tag (XML)	Enter an XML tag name to be used around the field. The default value is the name of the field. **Note.** Although each field name in a file record must be unique, each field tag does not have to be unique.

Strip Characters

Specify any characters to be removed from the input buffer. You use this to preprocess input strings. For example, if a field in your input file contains hyphens, but you want to remove the hyphens prior to processing the field, you could enter a hyphen here, and it would be stripped out while being read. You can specify more than one character to be stripped out. Be sure to not separate the strip characters. For example, the following strips out all hyphens and semi-colons:

```
-;
```

The following strips out all hyphens, semi-colons and spaces:

```
; -
```

Trim Spaces

Select to remove the leading and trailing spaces from the input string but leave spaces within the string intact.

This is different from the Strip Characters field, which will remove all spaces from the entire input field if you specify a space.

Field Description

Enter a description of the field for documentation purposes.

Field Inheritance

Optionally select a parent file record and field, from which the current field's value is to be inherited. If you're writing to a file, this means the value will only be written in the parent file record, not the child (inheriting) file record (that is, the value won't be written more than once to a file.) If no value is present in the parent field, the default value specified here will be used.

For example, the following file sample shows both the EMPLID (8113) and EFFDT (08/06/1999) written only once to a file, though these fields are repeated in the third file record (with File Record ID 102.)

```
100 8113 Frumman,Wolfgang
101      08/06/1999     000001   219 Going to London
office
102                100 000015 I 08/06/1999
102                200 000030 I 08/06/1999
102                300 000009 I 08/06/1999
102                400 000001 I 08/06/1999
102                500 000011 I 08/06/1999
```

See Also

PeopleTools PeopleBook: Application Designer, "Creating Field Definitions"

Performing Data Interchange

This section describes how to:

- Export data.
- Understand the data import process.
- Preview the input data.
- Generate and run the import program.

Exporting Data

The method you use to export data from PeopleSoft depends on the target application's requirements. To export data to a flat file, you'll create a file layout definition, then write PeopleCode to transfer the data to a file. The PeopleCode can be launched from Application Engine or any event. It should populate text strings, rowset objects or record objects, and apply the File class WriteRecord or WriteRowset method to transfer the data to the file, using the file layout definition to position the records and fields as required by the target application.

Note. To generate valid XML files, be sure to use the file class Close method when you finish writing to the file.

See Also

PeopleTools PeopleBook: PeopleCode Reference, "PeopleCode Classes," File Class

Understanding the Import Process

To help you troubleshoot and import flat file data, the file layout definition provides a data preview page. It can also generate an Application Engine program with associated PeopleCode necessary to import the data.

Data Import Activities

1. Provide the import data in a properly formatted flat file. Each record in the file must correspond to a record with the same name in your PeopleSoft database, and its fields must have the same names and data formats as the database record's fields. Each record in the file must end with a newline character.

2. Create a file layout definition to match the record and field structure of your data. Insert the appropriate record definitions into your file layout, then reposition the file records and file fields to match the record and field positions in your file.

3. Preview and troubleshoot the input data format and content.

4. Generate the data import Application Engine program and PeopleCode.

5. Run the Application Engine program to import the data.

Including and Excluding Fields

The fields in your data file's records can be a subset of the database record's fields — define your file layout with only the file fields you expect to receive.

The fields in your data file's records can be a superset of the database record's fields; you must define your file layout to suppress or ignore the extra fields. For FIXED files, don't define a field at the corresponding position in your file layout definition. For CSV files, the file layout must have the same number of fields in each record as there are in the corresponding file record — for each field you don't want to import, define a field in the file layout at that position that *doesn't* correspond to any field in that database record. For XML files, any extra fields are ignored automatically.

Your data file can contain a subset or a superset of the records defined in the file layout. Only a file record with a matching file record ID in the file layout is imported.

Record Hierarchy

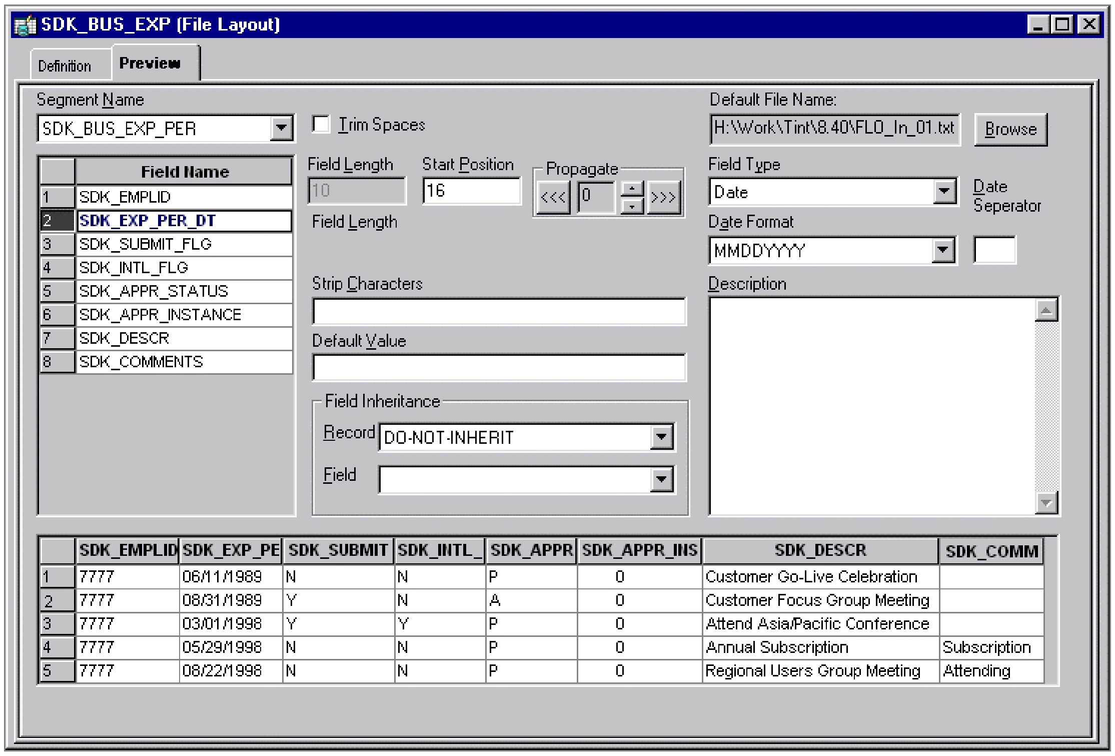
In theory, you can ignore rowset hierarchy when importing file data, because the PeopleSoft database stores each record independently of the others, and rowsets aren't used in the import process. However, many records are designed with hierarchical dependencies in mind. The input file might omit inherited field values or order the data records in a way that reflects such dependencies.

If your input file omits inherited field values, make sure the inheriting fields' records in the file layout are children of the ones from which they inherit their values, and make the appropriate Field Inheritance settings.

If the records to be imported contain key fields that reflect a rowset hierarchy, they might be in an order in the file that also reflects the hierarchy. Make sure your file layout reflects that hierarchy as well.

Important! Your completed file layout must have exactly one file record at the root level; all other file records must be below that level. See Reordering File Layout Nodes to learn how to move file records down.

Previewing Input Data


SDK_BUS_EXP (File Layout)

Definition | Preview

Segment Name: SDK_BUS_EXP_PER

☐ Trim Spaces

Default File Name: H:\Work\Tint\8.40\FLO_In_01.txt

Browse

	Field Name
1	SDK_EMPLID
2	SDK_EXP_PER_DT
3	SDK_SUBMIT_FLG
4	SDK_INTL_FLG
5	SDK_APPR_STATUS
6	SDK_APPR_INSTANCE
7	SDK_DESCR
8	SDK_COMMENTS

Field Length: 10

Start Position: 16

Propagate: 0

Field Type: Date

Date Seperator:

Date Format: MMDDYYYY

Strip Characters:

Default Value:

Description:

Field Inheritance

Record: DO-NOT-INHERIT

Field:

	SDK_EMPLID	SDK_EXP_PE	SDK_SUBMIT	SDK_INTL_	SDK_APPR	SDK_APPR_INS	SDK_DESCR	SDK_COMM
1	7777	06/11/1989	N	N	P	0	Customer Go-Live Celebration	
2	7777	08/31/1989	Y	N	A	0	Customer Focus Group Meeting	
3	7777	03/01/1998	Y	Y	P	0	Attend Asia/Pacific Conference	
4	7777	05/29/1998	N	N	P	0	Annual Subscription	Subscription
5	7777	08/22/1998	N	N	P	0	Regional Users Group Meeting	Attending

File layout preview

To preview the input data:

1. Open the appropriate file layout definition in Application Designer.

2. Go to the Preview tab.

   Initially no information is displayed.

   Note. With an XML file, the data can't be previewed, although you can still generate and run an import program. See Generating and Running the Import Program.

3. Click the Browse button and select your import file.

   The name of the import file appears in the Import File text box.

4. Select a file layout segment from the Segment dropdown list.

   The file layout fields corresponding to the selected segment appear in the list below the segment name. This list comes from the file layout definition, and does not depend on you selecting a valid import file.

5. Click the Refresh button on the toolbar to refresh the preview data.

At the bottom of the window, a preview of the first five rows of the selected segment in the selected import file appears in the grid.

You can also select View, Automatically Read Preview Data. While this menu item is selected, the file layout refreshes the preview automatically; when it's cleared, you need to click the Refresh Preview Data button to see the effects of changes you make to the property settings or data.

Note. Preview file data is available only for Fixed and CSV file layouts. The preview data only appears if you've selected an import file, if the file format matches the format specified for the file layout definition, and if it contains a file record ID that matches the file record ID of the selected segment.

6. Select one of the fields on the Field Name list.

   The properties of the field you selected appear in the appropriate display fields. These properties are actually the field's file layout field properties; you can change all field property settings from this view.

Note. Only the field properties appropriate to the file layout format you specified in the File Layout Definition Properties dialog box are visible in this view; for example, the Start Position and Field Length fields are available only for a FIXED format file layout.

Examining the Input Data for Errors

The following table shows some examples of input data errors.

Symptom	***Possible Reason***	***Solution***
The preview grid doesn't appear.	The input file's format doesn't match the file layout format you specified in the File Layout Definition Properties dialog box.	Change the file layout format to match the input file.
	A CSV file doesn't use the definition qualifier you specified in the File Layout Definition Properties dialog box.	Change the definition qualifier to match the one used in the input file.
Only the first column of the preview grid is populated.	A CSV file doesn't use the definition delimiter you specified in the File Layout Definition Properties dialog box.	Change the definition delimiter to match the one used in the input file.

Symptom	***Possible Reason***	***Solution***
The preview grid appears for some records, but not for others.	The file's record IDs for the missing records don't match their file record IDs specified in the File Layout Segment Properties dialog box.	Specify file record IDs that match the input file records.
Data for a field appears truncated in the preview grid.	With a FIXED file, the field length you specified in the File Layout Field Properties dialog box is too short to accommodate the field data.	Increase the field length to accommodate the input data.
A field appears to start in the middle of the data.	With a FIXED file, the start position you specified in the File Layout Field Properties dialog box is too great to include the start of the field data.	Decrease the start position and adjust the field length to match the input data start position and length.

Generating and Running the Import Program

To generate and run the import program:

1. After you preview your file layout and examined the input data, click the **AE** button on the toolbar.

   This will generate the Application Engine import program with its associated import PeopleCode.

2. Enter a name for the Application Engine program and click OK.

   The program is automatically saved and is ready to run as soon as its definition appears in Application Designer.

3. Click the Run Program button on the toolbar.

   The file data is imported into your database.

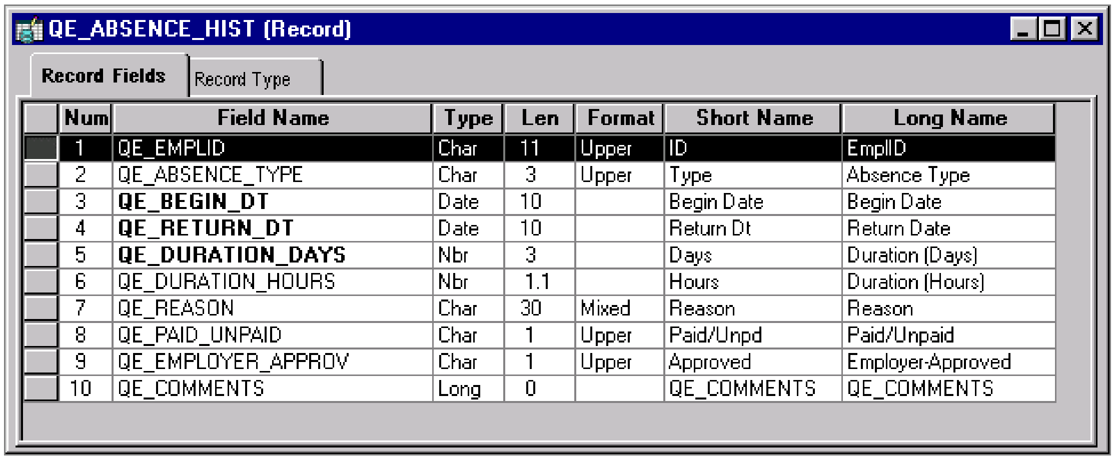
Creating a Sample File Layout

The following example illustrates creating a file layout that could be used with the QE_ABSENCE_HIST record.

This section describes how to:

- Create the file layout definition.
- Adjust the layout properties.

- Insert a segment and a file field.



QE_ABSENCE_HIST (Record)

Record Fields | Record Type

Num	Field Name	Type	Len	Format	Short Name	Long Name
1	QE_EMPLID	Char	11	Upper	ID	EmplID
2	QE_ABSENCE_TYPE	Char	3	Upper	Type	Absence Type
3	**QE_BEGIN_DT**	Date	10		Begin Date	Begin Date
4	**QE_RETURN_DT**	Date	10		Return Dt	Return Date
5	**QE_DURATION_DAYS**	Nbr	3		Days	Duration (Days)
6	QE_DURATION_HOURS	Nbr	1.1		Hours	Duration (Hours)
7	QE_REASON	Char	30	Mixed	Reason	Reason
8	QE_PAID_UNPAID	Char	1	Upper	Paid/Unpd	Paid/Unpaid
9	QE_EMPLOYER_APPROV	Char	1	Upper	Approved	Employer-Approved
10	QE_COMMENTS	Long	0		QE_COMMENTS	QE_COMMENTS

QE_ABSENCE_HIST record definition

For simplicity, let's say each row in our Fixed format data file has the following structure:

```
888 A

000 8001 VAC  1981-09-12 1981-09-26 14  .0                    P Y

888 A

000 8001 VAC  1983-03-02 1983-03-07 5   .0                    P Y

888 A

000 8001 VAC  1983-08-26 1983-09-10 13  .0                    P Y

888 A

000 8516 MAT  1986-06-06 1986-08-01 56  .0                    P Y

888 C

000 8516 SCK  1988-08-06 1988-08-07 1   .0                    P Y

888 A

000 8516 VAC  1987-07-14 1987-07-28 14  .0                    P Y

888 A

000 8553 JUR  1990-12-12 1990-12-17 5   .0  Local Jury Duty   P N

888 A

000 8553 MAT  1992-02-20 1992-10-01 224 .0  Maternity Leave   U N

888 A

000 8553 MAT  1994-08-19 1995-03-01 194 .0  Maternity         U Y

888 A

000 8553 PER  1993-04-15 1993-04-19 4   .0                    U N Personal
Day required

888 C

000 8553 SCK  1987-01-28 1987-01-30 2   .0  Hong Kong Flu     P  N

888 A

000 8553 SCK  1988-08-02 1988-08-03 1   .0  Sick              P  N

888 A

000 8553 SCK  1995-09-12 1995-09-13 1   .0                    P  N

888 C
```

```
000 G001 MAT  1991-07-02 1991-09-28 88  .0  3-month Maternity  P  Y Maternity
will be paid as 80% of Claudia's current salary.
```

000 is the file record ID for ABSENCE_HIST, and each field appears in the same order as in the ABSENCE_HIST database record. 888 is the file record ID for an extra segment called CHANGE_ACTION, containing an AUDIT_ACTION field with the following meanings:

- A: Row inserted.
- C: Row updated, but no key fields changed.

Note. The end of file (EOF) character must be on a separate line and not on a line containing data for any incoming file, regardless of file type. Each data line needs to be terminated with an end of line (EOL) character, which is different than an EOF.

Creating the Definition

To create a file layout definition:

1. Use the QE_ABSENCE_HIST record definition as a template for the file layout.

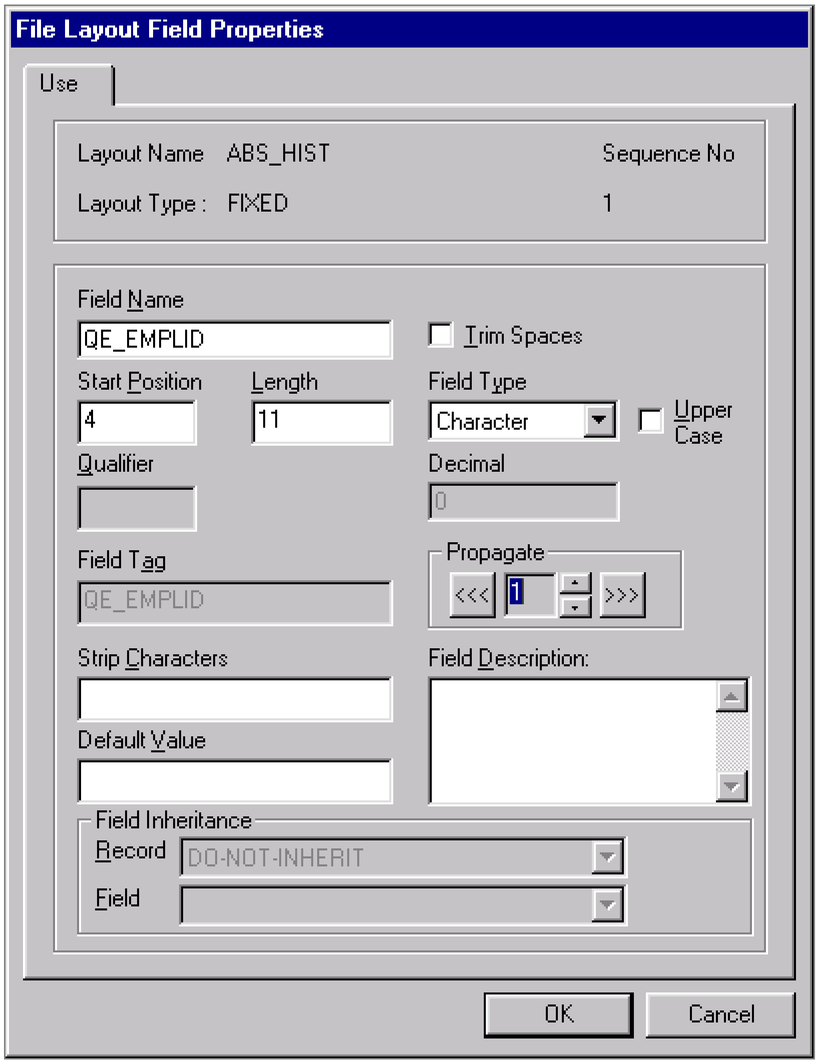
   Create a new file layout, then drag the QE_ABSENCE_HIST record into the open file layout.

2. Save the file layout.

   Save it with a name of ABS_HIST.

   The name of the first node changes from NEW FILE to ABS_HIST.

Adjusting Properties


File Layout Field Properties

Use

Layout Name ABS_HIST
Sequence No 1
Layout Type : FIXED

Field Name: QE_EMPLID
Trim Spaces: unchecked
Start Position: 4
Length: 11
Field Type: Character
Upper Case: unchecked
Qualifier:
Decimal: 0
Field Tag: QE_EMPLID
Propagate: 1
Strip Characters:
Field Description:
Default Value:
Field Inheritance
Record: DO-NOT-INHERIT
Field:

File Layout Field Properties example

To adjust layout, record, and field properties:

1. Change the file layout properties.

   Double-click the topmost node in the file layout, ABS_HIST, to display the file layout properties. Fill in a short and long description of the file layout you're creating. For this example, we're creating a FIXED file layout, so you don't need to make any changes on the Use tab.

2. Change the file record properties.

   Double-click the QE_ABSENCE_HIST file record to display its properties. Enter a record ID of 000, and a starting position of 1. The ID length is automatically set.

When you click OK, you'll get a message asking if you want to increment the start positions for all fields.

Click Yes. This will automatically increment the start position numbers for every field to take the length of the file Record ID you just added into account. If you don't click yes, you'll have to manually increment the start position for all your fields.

We've just created the file record ID for the QE_ABSENCE_HIST record:

```
000 8001       VAC  1981-09-12 1981-09-26 14  .0                      P Y
000 8001       VAC  1983-03-02 1983-03-07 5   .0                      P Y
```

3. Change the file field properties.

   When a record definition is used as a template for a file layout, the default starting position for each field is based on the order it appears in the record as well as its length.

   Double-click the QE_EMPLID file field to display its properties.

   The start position is automatically incremented to 4 (since the file record ID is three characters long). However, in the example there's an extra space between the end of the file record ID and the first field. Therefore, you need to change the start position of this field, and all the fields after this field. To do this:

   - Click the up arrow under Propagate to change that number from 0 to 1.
   - Click the button with the arrows pointing right (>>>).

   This increments the starting position of this field and all fields following this field by 1.

4. Adjust other fields (optional).

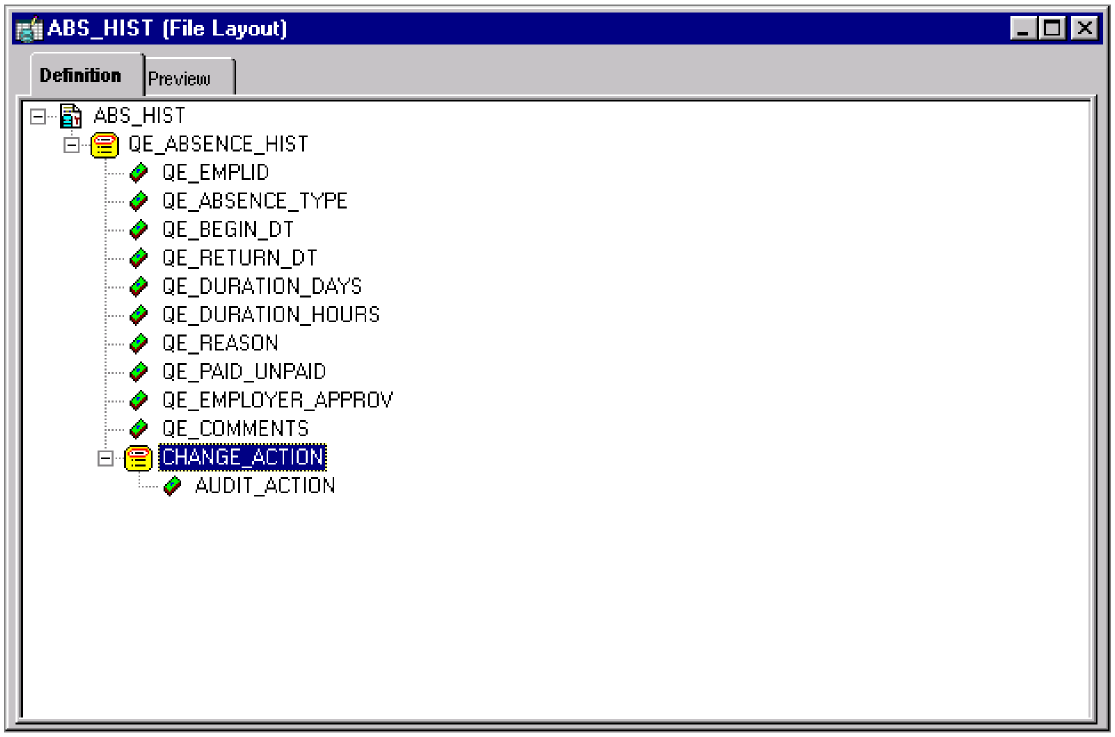
   The last field (QE_COMMENTS) has a length of 0. This is because it's based on a field of type Long.

Note. When a Long field is inserted into a file layout, it's converted to a Character field with a length of 0.

Because the format for this file layout is FIXED, you have to change the field length of that file field so it's long enough to accommodate the data you expect from the file.

You don't have to propagate this change because this is the last field in the record.

Inserting a Segment and Field



```
ABS_HIST
  QE_ABSENCE_HIST
    QE_EMPLID
    QE_ABSENCE_TYPE
    QE_BEGIN_DT
    QE_RETURN_DT
    QE_DURATION_DAYS
    QE_DURATION_HOURS
    QE_REASON
    QE_PAID_UNPAID
    QE_EMPLOYER_APPROV
    QE_COMMENTS
  CHANGE_ACTION
    AUDIT_ACTION
```

File layout with new segment and field added

To insert a segment and a field in that segment:

1. Insert a segment.

   We want to insert a segment that is a sibling, (that is, at the same level), as the QE_ABSENCE_HIST record. Insert a segment by selecting **Insert Segment** from the pop-up menu, or by going **Insert, Segment**. The Insert New Segment dialog box is displayed.

   This dialog box is identical to the File Layout Segment Properties dialog box. Fill in the File Record Name, File Record ID, and ID Start Position fields. When you click OK, the segment is inserted.

2. Insert a file field.

   Insert a file field by selecting Insert FileField from the pop-up menu. The Insert New Field dialog box appears. It's identical to the File Layout Field Properties dialog box. Fill in the Field Name, Start Position, and Field Length fields, then click OK.

   Note. The start position isn't automatically set when you add a file field to the file record; for this example the AUDIT_ACTION field requires a start position of 5.

3. Save your work.

Be sure to save the changes you've made to your file layout by going to File, Save, or clicking the save icon in the toolbar.

Now that you’ve created and saved a file layout, you must use PeopleCode to access the data. file layouts rely solely on PeopleCode as the engine behind the actual data access and movement.

See Also

PeopleTools PeopleBook: PeopleCode Reference, “PeopleCode Classes,” File Class

CHAPTER 2

The PeopleSoft API Repository

This chapter provides an overview of the PeopleSoft API Repository and describes:

- Repository examples.
- Repository properties.
- Bindings collection properties.
- Bindings collection methods.
- Bindings properties.
- Bindings methods.
- Namespaces collection properties.
- Namespaces collections methods.
- Namespaces properties.
- Namespaces methods.
- ClassInfo collection properties.
- ClassInfo collection methods.
- ClassInfo properties.
- MethodInfo collection properties.
- MethodInfo collection methods.
- MethodInfo properties.
- PropertyInfo collection properties.
- PropertyInfo collection methods.
- PropertyInfo properties.
- Summary of repository methods and properties.

Understanding the PeopleSoft API Repository

The PeopleSoft API Repository allows PeopleCode and third-party integrators to discover the internally available classes, methods, and properties provided by PeopleSoft for integration. The repository is useful to third-party integrators who integrate in a generic fashion: middleware providers, testing tool providers, and automated documentation providers.

The PeopleSoft API Repository is not a necessary interface for integrators who integrate at the business rule level, such as integration with an expense report, and so on. Those integrators should use PeopleSoft Component Interfaces or PeopleSoft Business Interlinks.

The repository describes available PeopleSoft APIs and provides mechanisms to determine the classes available in the API, the properties of each class, the methods of a class (along with the required parameters), and information concerning which *group* a class belongs to (known as a namespace).

The process of determining information about the API is known as *discovery*. Third-party integrators use information found through discovery to drive generic integration tools.

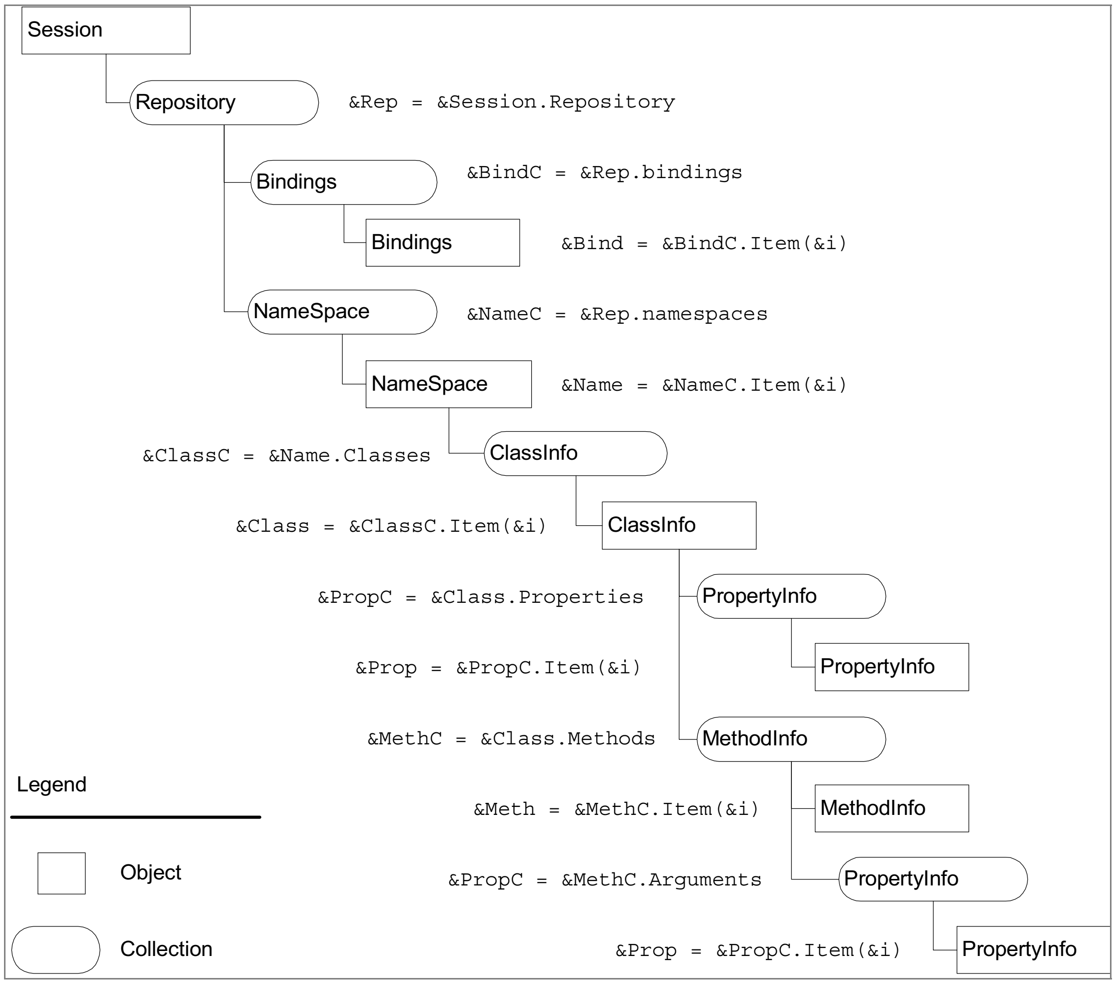
The repository is divided into namespaces. Each namespace contains a collection of related classes. Example namespaces include "PeopleSoft," "ComponentInterface," "Trees," and "BusinessInterlinks".

A *class* defines a related set of methods and properties. Using the repository, you can determine the methods and properties that are available and can be used on any object returned by a call to the PeopleSoft API. An *instance* of a class is known as an *object*.

A *property* is a data item of an object that has both a name and type (string, number, and Boolean, etc are examples of types). Some properties are used for inputting data to a class, some are used for getting data from a class, and some are used for both. Whether a property is used for input or output or both is known as *usage*.

A *method* is a function you can call on an object. Methods have a name and a return type (string, number, Boolean, and so on). Methods also have a collection of arguments that must be set prior to invoking the method. Methods arguments have identical attributes to properties.

The following diagram shows the different types of objects and collections instantiated from the repository:



Repository model

Repository Examples

This section describes:

- Retrieving information from the repository using PeopleCode.
- Understanding retrieval steps.
- Retrieving information from the repository using Visual Basic.

Retrieving Information From the Repository Using PeopleCode

The following example gets information for the class ABS_HIST from the Namespace component interface and writes it to the file BC.TXT

The following is the complete code sample, followed by the flat file. The next section presents steps that explain each line.

```
Local ApiObject &MYSESSION;
Local ApiObject &MYCI;
Local string &OutTEXT;
Local File &MYFILE;

&MYSESSION = GetSession();
&MYSESSION.Connect(1, "EXISTING", "", "", 0);

&MYFILE = GetFile("CI.txt", "A");

&NAMESPACES = &MYSESSION.Repository.Namespaces;
&NAMESPACE = &NAMESPACES.ItemByName("CompIntfc");

&OutTEXT = "Namespace = " | &NAMESPACE.Name;
&MYFILE.WriteLine(&OutTEXT);

&CLASSES = &NAMESPACE.classes;
&CLASS = &CLASSES.ItemByName("ABS_HIST");

&OutTEXT = "  Class: " | &CLASS.Name;
&MYFILE.WriteLine(&OutTEXT);

&OutTEXT = "     Methods";
&MYFILE.WriteLine(&OutTEXT);

&METHODS = &CLASS.methods;
For &K = 0 To &METHODS.Count - 1
   &METHOD = &METHODS.item(&K);
   &OutTEXT = "             " | &METHOD.name | ":  " | &METHOD.Type;
   &MYFILE.WriteLine(&OutTEXT);
   &ARGUMENTS = &METHOD.arguments;
   For &M = 0 To &ARGUMENTS.count - 1
      &ARGUMENT = &ARGUMENTS.item(&M);
      &OutTEXT = "           " | &ARGUMENT.name | ": " | &ARGUMENT.type;
      &MYFILE.WriteLine(&OutTEXT);
   End-For;
End-For;

&OutTEXT = "     Properties";
&MYFILE.WriteLine(&OutTEXT);

&PROPERTIES = &CLASS.properties;
For &I = 0 To &PROPERTIES.count - 1
   &PROPERTY = &PROPERTIES.item(&I);
   &OutTEXT = "              " | &PROPERTY.name | ": " | &PROPERTY.type;
```

```
     &MYFILE.WriteLine(&OutTEXT);
  End-For;
  &MYFILE.Close();
```

The above code produces the following flat file:

```
Namespace = CompIntfc
  Class: ABS_HIST
     Methods
             Get:  Boolean
             Save:  Boolean
             Cancel:  Boolean
             Find:  ABS_HIST
             GetPropertyByName:  Variant
           Name: String
             SetPropertyByName:  Number
           Name: String
           Value: Variant
             GetPropertyInfoByName:  CompIntfcPropertyInfo
           Name: String
     Properties
              EMPLID: String
              LAST_NAME_SRCH: String
              NAME: String
              ABSENCE_HIST: ABS_HIST_ABSENCE_HISTCollection
              interactiveMode: Boolean
              getHistoryItems: Boolean
              componentName: String
              compIntfcName: String
              stopOnFirstError: Boolean
              propertyInfoCollection: CompIntfcPropertyInfoCollection
              createKeyInfoCollection: CompIntfcPropertyInfoCollection
              getKeyInfoCollection: CompIntfcPropertyInfoCollection
              findKeyInfoCollection: CompIntfcPropertyInfoCollection
```

See Also

Example Using Visual Basic

Understanding Retrieval Steps

The following steps go through the code example line by line:

1. Get a session object.

Before you can access the PeopleSoft API Repository, you have to get a session object. The session controls access to PeopleSoft, provides error tracing, allows you to set the runtime environment, and so on.

```
&MYSESSION = GetSession();
&MYSESSION.Connect(1, "EXISTING", "", "", 0);
```

2. Open the file.

   As this text will be written to a flat file, the next step is to open the file. If the file is already created, the new text is appended to the end of it. If the file hasn't been created, the GetFile built-in function creates the file.

```
&MYFILE = GetFile("CI.txt", "A");
```

3. Get the namespace you want.

   Use the Namespaces property on the repository object to get a collection of available namespaces. We want to discover information about a component interface, so we specify CompIntfc in the ItemByName method to get that namespace. With ItemByName, you must specify a namespace that already exists. You'll receive a runtime error if you specify one that doesn't exist.

```
&NAMESPACES = &MYSESSION.Repository.Namespaces;
&NAMESPACE = &NAMESPACES.ItemByName("CompIntfc");
```

4. Write the text to the file.

   Because all the information discovered is being written to a file, the next step is to write text to the file. This code writes the string *Namespace,* followed by the name of the namespace, to the file.

```
&OutTEXT = "Namespace = " | &NAMESPACE.Name;
&MYFILE.WriteLine(&OutTEXT);
```

5. Get the class you want and write text to the file.

   Use the Classes property on the Namespace object to get a collection of all the available classes. We want to discover information about the component interface named ABS_HIST, so we specify that using ItemByName. Then we write that information to the file.

```
&CLASSES = &NAMESPACE.classes;
&CLASS = &CLASSES.ItemByName("ABS_HIST");

&OutTEXT = "  Class: " | &CLASS.Name;
&MYFILE.WriteLine(&OutTEXT);
```

6. Get the methods and arguments, and write the information to the file.

   Use the Methods property on the Class object to get a collection of all the available methods. After you get each method and write the information to the file, loop through and find all of the arguments for the method, then write that information to the file.

```
&OutTEXT = "     Methods";
&MYFILE.WriteLine(&OutTEXT);

&METHODS = &CLASS.methods;
For &K = 0 To &METHODS.Count - 1
   &METHOD = &METHODS.item(&K);
   &OutTEXT = "              " | &METHOD.name | ":  " | &METHOD.Type;
   &MYFILE.WriteLine(&OutTEXT);
   &ARGUMENTS = &METHOD.arguments;
   For &M = 0 To &ARGUMENTS.count - 1
      &ARGUMENT = &ARGUMENTS.item(&M);
      &OutTEXT = "           " | &ARGUMENT.name | ": " | &ARGUMENT.type;
      &MYFILE.WriteLine(&OutTEXT);
   End-For;
End-For;
```

7. Get the properties and write the information to the file.

Use the Properties property on the Class object to get a collection of all the available properties. Write each property, with its type, to the file. At the end of the program, close the file.

```
&OutTEXT = "     Properties";
&MYFILE.WriteLine(&OutTEXT);

&PROPERTIES = &CLASS.properties;
For &I = 0 To &PROPERTIES.count - 1
   &PROPERTY = &PROPERTIES.item(&I);
   &OutTEXT = "              " | &PROPERTY.name | ": " | &PROPERTY.type;
   &MYFILE.WriteLine(&OutTEXT);
End-For;

&MYFILE.Close();
```

Example Using Visual Basic

The following example gets information for the class ABS_HIST from the Namespace component interface.

```
Private Sub Command1_Click()
    '****************************************************************
    '* TacDemo: Example Repository Usage from Visual Basic
    '*
    '* Copyright (c) 1999 PeopleSoft, Inc.   All rights reserved.
    '****************************************************************

    ' Declare variables
    Dim oSession As New PeopleSoft_PeopleSoft.Session
    Dim oPSMessages As PSMessageCollection
    Dim oPSMessage As PSMessage

    ' Establish a PeopleSoft Session
    nStatus = oSession.Connect(1, "//PSOFTO070698:9001", "PTDMO", "PTDMO", 0)

    ' Enable error-handler
    On Error GoTo ErrorHandler

    ' Get a Component Interface "shell"
    Dim oNamespaces As NamespaceCollection
    Dim oNamespace As Namespace
    Dim oClasses As ClassInfoCollection
    Dim oClass As ClassInfo
    Dim oMethods As MethodInfoCollection
    Dim oMethod As MethodInfo
    Dim oArguments As PropertyInfoCollection
    Dim oArgument As PropertyInfo
    Dim oProperties As PropertyInfoCollection
    Dim oProperty As PropertyInfo

    Set oNamespaces = oSession.Repository.namespaces
    Set oNamespace = oNamespaces.ItemByName("ComponentInterface")

    Dim outText As String

    outText = "Namespace = " & oNamespace.Name & vbNewLine

    Set oClasses = oNamespace.classes
    Set oClass = oClasses.ItemByName("ABS_HIST")

    outText = outText & "   Class: " & oClass.Name & vbNewLine

    outText = outText & "       Methods" & vbNewLine

    Set oMethods = oClass.methods
    For k = 0 To oMethods.Count - 1
        Set oMethod = oMethods.Item(k)
        outText = outText & "           " & oMethod.Name & ":  " &
oMethod.Type & vbNewLine
        Set oArguments = oMethod.arguments
        For m = 0 To oArguments.Count - 1
            Set oArgument = oArguments.Item(m)
            outText = outText & "                 " & oArgument.Name & ":  " &
oArgument.Type & vbNewLine
```

```
        Next
    Next

    outText = outText & "       Properties" & vbNewLine

    Set oProperties = oClass.properties
    For k = 0 To oProperties.Count - 1
        Set oProperty = oProperties.Item(k)
        outText = outText & "            " & oProperty.Name & ":  " &
oProperty.Type & vbNewLine
    Next

 txtResults = outText


' Leave before we encounter the error handler
Exit Sub

ErrorHandler:
    If Err.Number = 1001 Then                ' PeopleSoft Error
        Set oPSMessages = oSession.PSMessages
        If oPSMessages.Count > 0 Then
            For i = 1 To oPSMessages.Count
               Set oPSMessage = oPSMessages.Item(i)
               MsgBox (oPSMessage.Text)
            Next i
            oPSMessages.DeleteAll
        Else
            MsgBox ("PS Api Error.  No additional information available from
Session log")
        End If
    Else                                    ' VB Error
        MsgBox ("VB Error: " & Err.Description)
    End If

End Sub
```

Repository Properties

Bindings

The Bindings property returns a reference to a Bindings collection.

This property is read-only.

Namespaces

The Namespaces property returns a reference to a Namespaces collection.

This property is read-only.

Bindings Collection Properties

Count

This property returns the number of Bindings Properties objects in the Bindings collection object.

Note. All repository counts begin at zero, not one.

This property is read-only.

Example

```
&COUNT = &BINDINGS.Count;
```

See Also

Bindings Properties

Bindings Collection Methods

Item

Syntax

`Item``(`*`number`*`)`

Description

The Item method returns a Bindings object that exists at the number position in the Bindings collection executing the method

Parameters

number	Specify the position number in the collection of the Bindings object that you want returned.

Returns

A reference to a Bindings object or NULL.

Example

```
For &N = 0 to &BINDINGS.Count - 1
     &BINDING = &BINDINGS.Item(&N);
     /* do processing */
End-For;
```

Bindings Properties

Name

This property returns the name of the object as a string.

This property is read-only.

Bindings Methods

Generate

Syntax

```
Generate()
```

Description

This method is a reserved internal function and shouldn't be used at this time.

Namespaces Collection Properties

Count

This property returns the number of Namespaces Properties objects in the Namespaces collection object.

Note. All repository counts begin at zero, not one.

This property is read-only.

Example

```
&COUNT = &NameC.Count;
```

See Also

Namespaces Properties

Namespaces Collections Methods

Item

Syntax

```
Item(number)
```

Description

The Item method returns a Namespaces object that exists at the *number* position in the Namespaces collection executing the method.

Parameters

number	Specify the position number in the collection of the Namespaces object that you want returned.

Returns

A reference to a Namespaces object or NULL.

Example

```
For &N = 0 to &NAMESPACES.Count - 1
     &NAMESPACE = &NAMESPACES.Item(&N);
     /* do processing */
End-For;
```

ItemByName

Syntax

```
ItemByName(name)
```

Description

The ItemByName method returns the item specified by *name*. *Name* is not case-sensitive.

Parameters

name	Specify the name of the Namespaces object that you want returned. This parameter takes a string value.

Returns

A reference to a Namespaces object or NULL.

Example

```
&NAMESPACE = &NAMESPACES.ItemByName("BusinessComponent");
```

Namespaces Properties

Classes

This property returns a reference to a ClassInfo Collection Properties collection.

This property is read-only.

Example

```
&CLASSC = &NAME.Classes;
```

See Also

ClassInfo Collection Properties

Name

This property returns the name of the object as a string.

This property is read-only.

Namespaces Methods

CreateObject

Syntax

`CreateObject``(`*`classname`*`)`

Description

This method is a reserved internal function and shouldn’t be used at this time.

ClassInfo Collection Properties

Count

This property returns the number of ClassInfo Properties objects in the ClassInfo collection object.

Note. All repository counts begin at zero, not one.

This property is read-only.

Example

```
&COUNT = &InfoC.Count;
```

See Also

ClassInfo Properties

ClassInfo Collection Methods

Item

Syntax

`Item``(`*`number`*`)`

Description

The Item method returns a ClassInfo object that exists at the *number* position in the ClassInfo collection executing the method.

Parameters

number	Specify the position number in the collection of the ClassInfo object that you want returned.

Returns

A reference to a ClassInfo object or NULL.

Example

```
For &N = 0 to &CLASSES.Count - 1
     &CLASS = &CLASSES.Item(&N);
     /* do processing */
End-For;
```

ItemByName

Syntax

`ItemByName``(`*`name`*`)`

Description

The ItemByName method returns the item specified by *name*. *Name* is not case-sensitive.

Parameters

name	Specify the name of the ClassInfo object that you want returned. This parameter takes a string value.

Returns

A reference to a ClassInfo object or NULL.

Example

```
&CLASS = &CLASSES.ItemByName("ABS_HIST");
```

ClassInfo Properties

Documentation

This property doesn't actually return all the documentation for the class, just a brief description of the class as a string.

This property is read-only.

Methods

This property returns a reference to a MethodInfo Collection Methods collection.

This property is read-only.

See Also

MethodInfo Collection Methods

Name

This property returns the name of the object as a string.

This property is read-only.

Properties

This property returns a reference to a PropertyInfo Collection Methods collection.

This property is read-only.

See Also

PropertyInfo Collection Methods

MethodInfo Collection Methods

Item

Syntax

`Item``(`*`number`*`)`

Description

The Item method returns a MethodInfo object that exists at the *number* position in the MethodInfo collection executing the method.

Parameters

number	Specify the position number in the collection of the MethodInfo object that you want returned.

Returns

A reference to a MethodInfo object or NULL.

Example

```
For &K = 0 To &METHODS.Count - 1
   &METHOD = &METHODS.item(&K);
```

```
      &OutTEXT = "             " | &METHOD.name | ":  " | &METHOD.Type;
      &MYFILE.WriteLine(&OutTEXT);
   End-For;
```

ItemByName

Syntax

```
ItemByName(name)
```

Description

The ItemByName method returns the item specified by *name*. *Name* is not case-sensitive.

Parameters

name	Specify the name of the MethodInfo object that you want returned. This parameter takes a string value.

Returns

A reference to a MethodInfo object or NULL.

Example

```
&METHOD = &METHODS.ItemByName("Save");
```

MethodInfo Collection Properties

Count

This property returns the number of MethodInfo Properties objects in the MethodInfo collection object.

Note. All repository counts begin at zero, not one.

This property is read-only.

Example

```
&COUNT = &MethC.Count;
```

See Also

MethodInfo Properties

MethodInfo Properties

Arguments

This property returns a reference to a PropertyInfo Collection Methods collection.

This property is read-only.

See Also

PropertyInfo Collection Methods

Documentation

This property doesn't actually return all the documentation for the class, just a brief description of the class, as a string.

This property is read-only.

Name

This property returns the name of the object as a string.

This property is read-only.

Type

This property returns the type of the method. Valid values include:

- Bool (Boolean).
- Number.
- Float.
- String.
- Variant.
- Blob (Binary large object).
- Any API class name.

This property is read-only.

PropertyInfo Collection Methods

Item

Syntax

```
Item(number)
```

Description

The Item method returns a PropertyInfo object that exists at the *number* position in the PropertyInfo collection executing the method.

Parameters

number	Specify the position number in the collection of the PropertyInfo object that you want returned.

Returns

A reference to a PropertyInfo object or NULL.

Example

```
For &K = 0 To &PROPERTIES.Count - 1
   &PROPERTY = &PROPERTIES.item(&K);
   &OutTEXT = "             " | &PROPERTY.name | ":  " | &PROPERTY.Type;
   &MYFILE.WriteLine(&OutTEXT);
End-For;
```

ItemByName

Syntax

```
ItemByName(name)
```

Description

The ItemByName method returns the item specified by *name*. *Name* is not case-sensitive.

Parameters

name	Specify the name of the PropertyInfo object that you want returned. This parameter takes a string value.

Returns

A reference to a PropertyInfo object or NULL.

Example

```
&PROPERTY = &PROPERTIES.ItemByName("GetHistoryItems");
```

PropertyInfo Collection Properties

Count

This property returns the number of PropertyInfo Properties objects in the PropertyInfo collection object.

Note. All repository counts begin at zero, not one.

This property is read-only.

Example

```
&COUNT = &PropC.Count;
```

See Also

PropertyInfo Properties

PropertyInfo Properties

Documentation

This property doesn't actually return all the documentation for the class, just a brief description of the class, as a string. This property is read-only.

Name

This property returns the name of the object as a string.

This property is read-only.

Type

This property returns the data type. Valid values are:

- Bool (Boolean).
- Number.
- Float.
- String.
- Variant.
- Blob (Binary large object).
- Any API class name.

This property is read-only.

Usage

This property returns a number that describes in which direction the specified property (or argument) can be passed. The following table describes the valid values.

Value	***Description***
0	Can be passed into PeopleSoft API.
1	Can be passed out of PeopleSoft API.
2	Can be passed either into or out of PeopleSoft API.

This property is read-only.

Summary of Repository Methods and Properties

Summary of Repository Methods

Repository objects are instantiated from the Repository property on a session object. This table contains a list of all the Repository objects plus their methods. Methods that can be used by a class are marked with an *X*.

Method	***Bindings collection***	***Namespaces collection***	***ClassInfo collection***	***MethodInfo collection***	***PropertyInfo collection***
CreateObject(cla ssname)		**X**			
Generate()	**X**				

Method	***Bindings collection***	***Namespaces collection***	***ClassInfo collection***	***MethodInfo collection***	***PropertyInfo collection***
Item(number)	**X**	**X**	**X**	**X**	**X**
ItemByName(name)		**X**	**X**	**X**	**X**

Summary of Repository Properties

Repository collection objects are instantiated from the Repository property on a session object. This table contains a list of all the Repository objects plus their properties (marked with an *RO* in the table). All properties are read-only.

Property	***Rep coll/ object***	***Binding coll/ object***	***Namespace coll/ object***	***ClassInfo coll/ object***	***MethodInfo coll/ object***	***PropertyInfo coll/ object***
Arguments					**RO**	
Bindings	**RO**					
Classes			**RO**			
Count		**RO**	**RO**	**RO**	**RO**	**RO**
Documentation				**RO**	**RO**	**RO**
Generate		**RO**				
Methods				**RO**		
Name		**RO**	**RO**	**RO**	**RO**	**RO**
Namespaces	**RO**					
Properties				**RO**		
Type					**RO**	**RO**
Usage						**RO**

Index