SIEBEL eBusiness 7

UPGRADING AND MIGRATING SIEBEL JANNA APPLICATIONS

VERSION 7.0, REV B

12-BCID6K

JULY 2002

Contents

Introduction

This guide is intended for those who have experience with Siebel Janna customizations using Siebel Janna Tools (formerly called Janna SDK) and Siebel configurations using Siebel Tools. Specific job roles of those that may be involved in the upgrade follow.

Depending on the product options that you have purchased, you should use this guide together with one or more of the following guides:

- *Siebel eFinance Guide*
- *Siebel Financial Services Guide*

Although job titles and duties at your company may differ from those listed in the following table, the audience for this guide consists of employees in these categories:

Application Architects	Persons responsible for the design and integration of enterprise software applications.
Database Administrators	Persons who administer the database system, including data loading; system monitoring, backup, and recovery; space allocation and sizing; and user account management.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.
Siebel Application Developers or Configurators	Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.

It is recommended that users of this guide be familiar with programming in Visual C++ and object-oriented design. Working knowledge of both Siebel Janna Tools, Siebel Tools, and Siebel eBusiness Application Integration is also beneficial.

How This Guide Is Organized

This guide is organized into chapters that outline the different tools and procedures required to perform the following migration strategies:

Strategy 1: Full Migration. This approach replaces the existing Siebel Janna application in favor of the Siebel 7 application. This strategy requires a comprehensive migration from the Siebel Janna database to a Siebel 7 supported database. The overview for doing so is documented in Chapter 1, "Database Upgrade Tools."

Strategy 2: Interoperability. This approach deploys the Siebel 7 application and integrates it with the existing Siebel Janna application allowing the use of existing business logic, data structures, and user interface components. In this scenario Siebel Janna behaves as a legacy application providing data and functionality to the Siebel 7 product. The framework and components involved are discussed in Chapter 2, "Interoperability." Once the theoretical foundations are covered, a hypothetical scenario is considered in Chapter 3, "Interoperability Case Study," which serves as a practical basis for existing implementations using this migration strategy.

This guide concludes with two appendixes. Appendix , "Siebel Janna Data Objects," lists specific Siebel Janna mapping information related to Chapter 2. The second appendix, Appendix , "Siebel Janna Terminology and Architecture," provides background information on Siebel Janna terminology and architecture.

There are many Siebel Janna features that have already been built into Siebel Financial Services. A discussion of these features is not within the scope of this document. For a list of new features that are available out-of-the-box, refer to *Siebel Financial Services Guide*.

This guide is to be used in combination with a broader documentation set that pertains to all Siebel eBusiness Applications. Most chapters in this guide cover only information that is specific to upgrading and migrating a Siebel Janna application. Where applicable, you are referred to other guides in the *Siebel Bookshelf* CD-ROM or other third-party documentation.

Using the Siebel Product

It is strongly recommended that you read *Fundamentals* so that you can make optimal use of your Siebel application, especially if you are new to Siebel software. *Fundamentals* provides detailed coverage of the Siebel user interface and how to use it; working with data; locating information with the query and find features; sharing information with other users; and so on. The features presented in *Fundamentals* appear throughout the Siebel application suite; they are introduced through procedures you can learn and use in your own Siebel application.

Revision History

Upgrading and Migrating Siebel Janna Applications, Version 7.0, Rev. B

Database Upgrade Tools 1

This chapter provides an overview of the requirements and tools necessary to upgrade the data side of a Siebel Janna application to a Siebel application. The core Siebel Janna base tables, which contain relevant customer, partner, and employee data, are migrated to a Siebel database using preconfigured third-party data migration utilities. The following section describes an overview of the utilities required for the database upgrade.

NOTE: The Database Upgrade Tools apply only to Siebel Janna for Institutional Finance and Siebel Janna Contact Enterprise. Upgrade earlier versions of the Janna product to one of Siebel Janna for Institutional Finance or Siebel Janna Contact Enterprise before upgrading to Siebel Financial Services 7.

Siebel Janna Database Upgrade Overview

The Siebel Janna Database Upgrade comprises the three basic steps shown in Figure 1.

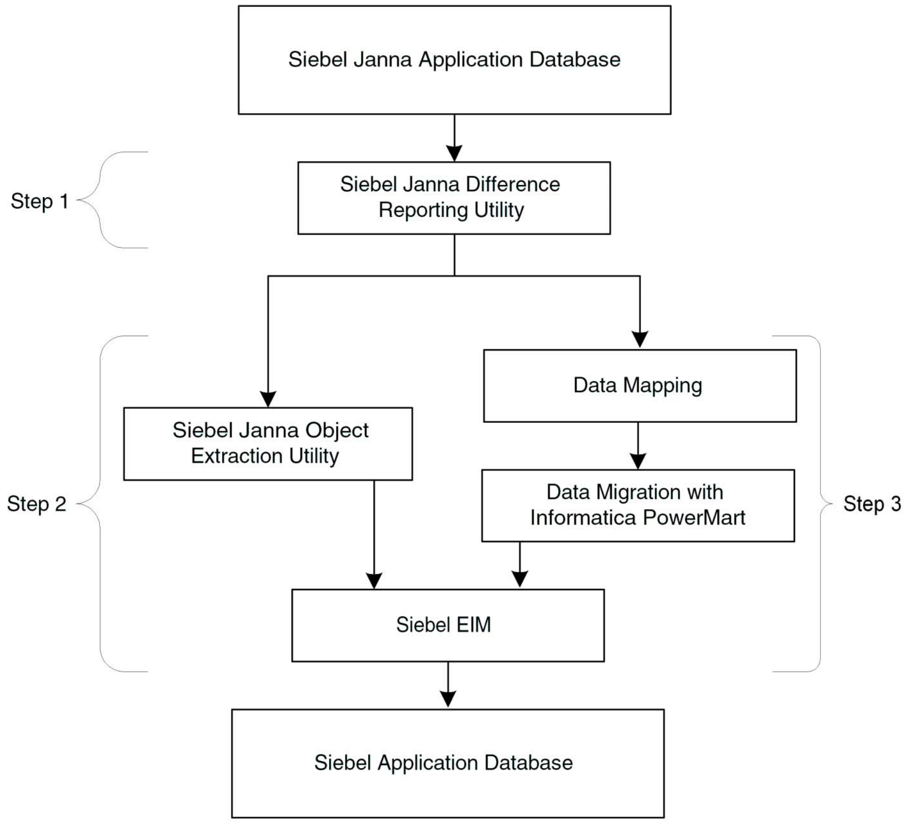

Siebel Janna Application Database
Step 1
Siebel Janna Difference Reporting Utility
Data Mapping
Siebel Janna Object Extraction Utility
Step 2
Data Migration with Informatica PowerMart
Step 3
Siebel EIM
Siebel Application Database

Figure 1. Siebel Janna Database Upgrade Overview

The rest of this chapter discusses each of the three steps in greater detail.

1. Document the differences between the standard Siebel Janna database and the configuration and usage of the client's database.

   Use the Siebel Janna Difference Reporting Utility for this procedure to produce a spreadsheet file with the difference information.

2. Extract attached objects (Binary Large Objects or BLOBs).

   The Siebel Janna application and the Siebel application allows users to associate spreadsheets, word processing documents, pictures, and similar data with accounts and contacts. These attached objects are upgraded using Siebel Enterprise Integration Manager (Siebel EIM), but first they must be converted to their native file format and placed in a staging area. The Siebel Janna Object Extraction Utility performs this operation. For further information on Siebel EIM, see *Siebel Enterprise Integration Manager Administration Guide*.

3. Map the schema and migrate the data through upgrade interface tables.

   This process involves mapping data elements in the Siebel Janna data model to functionally equivalent elements in the Siebel data model. Data elements for each available Siebel Janna product are mapped to available Siebel products. The data is extracted from the Siebel Janna database into Siebel EIM Interface Tables using Informatica's PowerMart utility. Siebel EIM then imports the data to the Siebel database.For further information on Siebel EIM, see *Siebel Enterprise Integration Manager Administration Guide*.

NOTE: It is recommended that data cleanup be performed prior to these three steps to make sure the upgrade process avoids any data loss.

Database Upgrade Tools

This section contains detailed information and procedures for the three database upgrade utilities.

Step 1: Siebel Janna Difference Reporting Utility

Each upgrade project requires customized Informatica mappings based on extension tables and certain lists of values (LOVs). To discover the necessary configurations that must be made to these mappings, run the Siebel Janna Difference Reporting Utility. This utility compares the base Siebel Janna tables out-of-the-box against the customized installation deployed at each specific client. It also captures a compilation of the changes to the types of Categories, Custom Fields, Phone Numbers, and Addresses.

While the Siebel Janna Difference Reporting Utility is running, it analyzes the Siebel Janna schema and, at its conclusion, produces a Microsoft Excel spreadsheet file with the difference information. The Excel spreadsheet is used to create further Informatica mappings for custom client tables and special usage of Siebel Janna fields, such as Contact.BaseDataString1. Non-system and non-core or module tables are listed as custom.

NOTE: Make sure that a Siebel Janna ODBC connection is available on the computer that the program is run on, and make sure that Microsoft Excel 97 or later is on this machine.

Installing the Siebel Janna Difference Reporting Utility

- Install the Siebel Janna Difference Reporting Utility on the workstation with the Siebel Janna ODBC connection by running the setup.exe program.

Running the Siebel Janna Difference Reporting Utility

To run the Siebel Janna Difference Reporting Utility

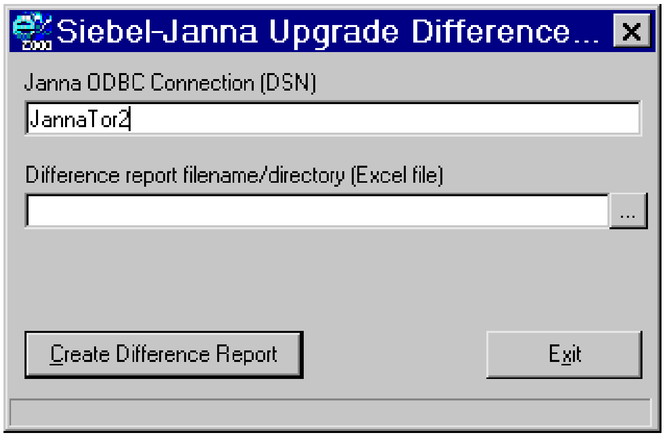
1. After a successful installation, run the DRT.exe file.

   The Siebel Janna Difference Reporting Utility window appears.

2 Enter the appropriate information in the two text fields.

The Siebel Janna ODBC Connection text box is, by default, filled with the UserDSN value from the registry.

The Difference report filename text box is used for the name and directory structure of the Excel spreadsheet file. Selecting the ellipsis button to the right of the field opens the Microsoft Save As dialog box where you can select or enter this information.


Siebel-Janna Upgrade Difference...

Janna ODBC Connection (DSN)

JannaTor2

Difference report filename/directory (Excel file)

3 Select the Create Difference Report button to create the Excel spreadsheet report.

While the report is prepared, the user sees a progress bar.

Step 2: Siebel Janna Upgrade Object Extraction Utility

Both Siebel Janna applications and Siebel applications allow users to associate spreadsheets, word processing documents, pictures, and other files with accounts and contacts. Siebel applications store pointers to files in their native format in a specific location in the file system. Siebel Janna applications also have this ability, but include another feature that stores these files as Binary Large Objects (BLOBs) in the database. The Siebel Janna Upgrade Object Extraction Utility migrates this file information to the Siebel database and is an executable file. To upgrade these objects saved within the Siebel Janna database, they must be extracted from their current state (BLOBs within the Siebel Janna database) and reconstituted into their original files. The extraction utility provides this process. Before using this utility, create new database tables using appropriate scripts.

Installing the Siebel Janna Upgrade Object Extraction Utility

- Install the Siebel Janna Upgrade Object Extraction Utility on the computer that contains the Siebel Janna Windows client by running the setup.exe program.

Creating New Database Tables

- Run the platform-specific SQL script—JUpg_<platform>.sql—against the Siebel Janna database to create the new database tables.

Running the Siebel Janna Upgrade Object Extraction Utility

To run the Siebel Janna Upgrade Object Extraction Utility

1. Run the ObjExtract.exe file on the appropriate computer.

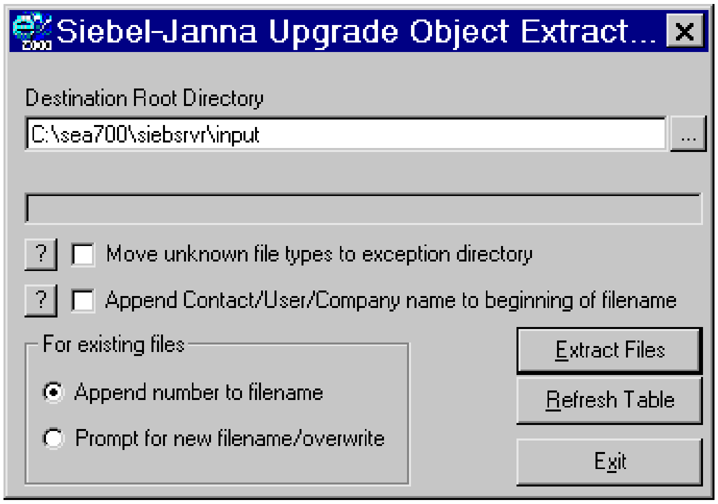
   The Siebel Janna Migration Object Extraction Utility window appears.

2. Enter the location of the destination root directory.

   This location can be entered manually or by clicking the ellipsis button and selecting a folder from the Set Folder window.

3 Select the existing file handling options.

Users have the option to rename existing file names manually or to have them automatically renamed (by adding sequential numbers to the existing filename).



Siebel-Janna Upgrade Object Extract...

Destination Root Directory: C:\sea700\siebsrvr\input

☐ Move unknown file types to exception directory

☐ Append Contact/User/Company name to beginning of filename

For existing files

(● Append number to filename) ○ Prompt for new filename/overwrite

- Optionally, check the *Move unknown file types to exception directory* check box, which moves any files whose extensions cannot be determined into an exception directory. If not checked, the files remain in the root directory specified. The ? button to the immediate left of the check box provides further information on this selection.
- Optionally, check the *Append Contact/User/Company name to beginning of filename* check box, which results in fewer files requiring renaming. However, selecting this option creates a separate copy of the same file for each Contact/User/Company associated with it. The ? button to the immediate left of the check box provides further information on this selection.

4 Click Extract File to begin the extraction procedure.

The files are then pulled from the Siebel Janna database and copied to the file system. Appropriate feedback is provided to the user during the conversion procedure and on completion two log files are produced: a success log, ObjSuccessLog.txt and an exception log, ObjExceptionLog.txt.

The Siebel Janna Upgrade Object Extraction Utility populates a table with the files extracted for each user. It creates an exception report log file for any errors or for any files that were extracted without extensions. Any files created without extensions are moved into a temporary subfolder under the root extraction directory. The extensions of these files must be manually added before the table is updated through the Refresh Table button.

Users can fix any filenames without extensions and then run the Siebel Janna Upgrade Object Extraction Utility again, this time pressing the Refresh Table button. This button updates the internal table used for the Siebel import with the names of the files that were manually changed.

Folders and subfolders are not carried forward to Siebel from the Siebel Janna Document Manager or Siebel Janna Contact Journal. Instead, the filename contains the folder path up to a total length (path and filename) of 200 characters—folder names are separated by a hyphen (-). If the total length is greater than 200 characters, the right end of the folder path is truncated. Multiple contact documents with attachments are converted so the attached document is duplicated for all contacts.

Object Extraction Utility Delta Dialog

If the Object Extraction Utility has already run, a Delta dialog appears. To run the entire object extraction again, select No. If you are running a subsequent update and want to only extract the files that have changed since the last time you ran the Object Extraction Utility, then select Yes.

NOTE: Perform an update only if you have already run the Object Extraction Utility and migrated the files to the Siebel file system. Running the update deletes the files in the extraction directory automatically, as only the new files added since the initial extraction will be imported to the Siebel file system. For the same reason, the update also deletes the existing records in the database table created to store the files.

Step 3: Schema Integration

The database upgrade is based on a logical Siebel Janna database. Prior versions need to upgrade to Siebel Janna using standard Siebel Janna upgrade scripts before the upgrade to a Siebel database can begin.

The database schema integration requires:

- Mapping the data elements from the Siebel Janna database to a Siebel database
- Extracting the data using Informatica's data migration utility, PowerMart
- Using Siebel EIM to import the data into the Siebel database

You can also upgrade client-specific custom extension tables by following a methodology similar to that used for the base tables—requirements and differences, data mapping, PowerMart, and Siebel EIM.

Installing the Siebel Janna Upgrade Sample Informatica Repository

This procedure assumes that Informatica PowerMart has been installed, in particular the client software.

To install the Siebel Janna Upgrade Sample Informatica Repository

1. Start Informatica Repository Manager.
2. Use the Restore Repository feature to install the sample repository file (Siebel_Janna_Migration.rep) to a database of your choice. Name the repository, Siebel_Janna_Migration, when prompted.
3. Run the Create.sql and Insert.sql scripts applicable to your database platform to create the Siebel Janna tables required by the sample mappings. Note that these scripts do not create or populate the tables required by the Object Extraction Utility (See "Step 2: Siebel Janna Upgrade Object Extraction Utility" on page 18 for further information).

Mapping Data Elements

Data elements in the Siebel Janna data model must be mapped to functionally equivalent elements in the Siebel data model. Basic data elements include information about Users, Contacts, Accounts, Interactions, and so forth. Details on the Siebel Janna elements are found in Appendix , "Siebel Janna Data Objects." The difference report from "Step 1: Siebel Janna Difference Reporting Utility" on page 16 reveals the differences between the Siebel Janna base tables and any client configuration. New tables and types are identified with this report and assist with mapping the data. Other mapping considerations follow:

- Siebel Janna Document Manager records are upgraded to the Siebel Literature module. The names of the upgraded documents in the Siebel Literature Module are made up of the fully qualified folder name and document name of the document in Siebel Janna.
- Records are assigned to the default organization. Implement organizational hierarchy after the data migration by creating business units and divisions. The Siebel Janna branch field is mapped to the NAME column in the S_ORG_EXT table to make this information visible. However, the company/branch hierarchy has been maintained at the database level.
- Where possible or required, map International Address Attributes (for example, House Number) stored in Siebel Janna to the appropriate Siebel Data Model.
- Based on specific needs, map data elements from Siebel Janna Custom Fields, Categories, or Document Custom Fields to similar fields in the Siebel data model, such as Region, Birthday, Territory, Spouse's Name, Dun & Bradstreet ID, and so on. Currently, this data is mapped to extension tables and columns so that there is no data loss.
- Siebel Janna remote rules are not migrated to the Siebel deployment.

Extracting Data

The extraction of data is implemented using a third-party data-migration utility, Informatica’s PowerMart. Core tables are mapped with standard mappings. Custom fields are manually mapped for each client as part of the Database Upgrade Methodology. Extracted data from Siebel Janna database is moved with the utility to Siebel EIM Interface tables. Siebel EIM is then used to import the data to the Siebel database.

Caution: The upgrade is one way and overwrites existing and previously upgraded Siebel data. After you start using the Siebel database, for production data, you cannot synchronize data back to the Siebel Janna database.

You must maintain functionally equivalent security to that of Siebel Janna applications after the upgrade. However, Siebel Janna record-level security data is not migrated. The security process is manually configured.

Using Siebel Enterprise Integration Manager (Siebel EIM)

Siebel EIM is the Siebel utility used to move data from interface tables to the Siebel database. Extracted data from the Siebel Janna database is moved with Informatica’s PowerMart utility to the interface tables used by Siebel EIM. For detailed information on Siebel EIM, see *Siebel Enterprise Integration Manager Administration Guide*.

A Siebel EIM configuration script is required, and provided, to import data elements from the interface tables.

Upgrade Considerations

Before completing the database upgrade, review the following considerations:

- Make sure Multiple User Action Items are covered.
- Make sure Multiple Contact Action Items are covered.
- Make sure international addresses are covered.
- Make sure data found in the JCSCode table is upgraded, if required.
- Make sure users set their time zone when logging into the system.
- The destination database must have the appropriate base currency configured before loading to make sure the currency values from Siebel Janna are properly covered.
- Date and time fields are stored in GMT in the Siebel Janna database and are migrated as such to the Siebel EIM Interface Tables.

Interoperability 2

This chapter provides an overview of Siebel Janna Interoperability and the potential approaches for integration implementation: data integration and user interface integration.

- **Data integration.** Uses Siebel eBusiness Application Integration (eAI) component methods to display and access data from an existing external Siebel Janna application.
- **User interface integration.** Displays Siebel Janna User Interfaces within the Siebel Financial Services Application, using an existing external Siebel Janna framework.

For background information on the Siebel Janna application architecture and terminology associated with Siebel Janna applications, see Appendix B, "Siebel Janna Terminology and Architecture."

For background information on the Siebel application architecture and integration procedures see *Siebel Tools Reference* and *Overview: Siebel eBusiness Application Integration Volume I* on the *Siebel Bookshelf*.

Integrating Data

In a Data Integration approach, the data is hosted by and resides in the existing Siebel Janna database. Access to the data is provided through a Siebel Business Service that communicates with the Siebel Janna Application Server (SJAS) in a distributed environment. Siebel Virtual Business Components will then use methods provided by this Business Service to populate Siebel Applets. Finally, Siebel Views containing these applets will be added to Siebel Screens that ultimately display and allow interaction with the Siebel Janna data.

Siebel Janna Virtual Business Component

The Siebel Janna Virtual Business Component (VBC) is similar to a Siebel Business Component in that it provides access to a set of data, but differs in that the data comes from an external data source, rather than the Siebel database. It is configured using user parameters at design time that specify the:

- Siebel Janna Gateway Business Service
- Siebel Janna Data Object
- Siebel Janna Service Object
- Other information required by the SJGBS

The Siebel Janna VBC interacts with the SJAS by invoking methods of the SJGBS.

NOTE: A separate Siebel Janna VBC is required for each SJDO whose data needs to be accessed.

Siebel Janna Gateway Business Service

The Siebel Janna Gateway Business Service (SJGBS) is a prebuilt Siebel Business Service hosted by the Siebel Object Manager. SJGBS makes requests on behalf of the Siebel Financial Services application to SJAS. It supports two types of Siebel Janna Service Object invocation:

- **SJAS XML Siebel Janna Service Object Call.** An XML document is formed that fully describes the method invocation. This XML document is submitted to the SJAS, which returns a response in the form of an XML document. This style of method invocation supports DCOM, HTTP, or MQSeries as transports. This makes it possible to integrate SJAS with UNIX-based Siebel Servers.
- **DCOM Siebel Janna Service Object Call.** A Siebel Janna Service Object is instantiated on the SJAS, and it receives method calls using the normal DCOM marshalling process. This process only supports DCOM as a transport, and therefore, requires that the Siebel Server executing this business service be run on Windows NT.

The SJGBS is also responsible for receiving the results of the method invocations and translating them into the appropriate Integration Objects. Integration Objects based on Siebel Janna Data Object XML DTDs will need to be mapped to the appropriate Siebel Janna Service Object, so that requests to the SJAS are for the appropriate service for each Siebel Janna Data Object. The SJGBS will resolve the operation to the correct Siebel Janna Service Object for activation inside the SJAS. The SJGBS implements the following seven methods:

- Delete
- Insert
- Init
- PreInsert
- Query
- Update
- Execute

These seven methods are analogous to the following Siebel Janna Service Object Interface methods:

Delete	IJGenObject.Delete
Insert	IJGenObject.AddUpdate
Init	No direct analogy, may translate to some necessary prework, like using the Integration Object definition for a VBC's initial list of fields
PreInsert	No direct analogy, may translate to some necessary prework, like generating a Siebel Janna Key for the Integration ID Parameter
Query	IJGenObject.Read
Update	IJGenObject.AddUpdate
Execute	Comprehensive method for utility and other functions, possibly invoking a Siebel Janna Service Object DoWork method

The SJGBS forms method requests (either using XML or DCOM) from supplied parameters (Integration Objects, other information) and submits requests to the SJAS.

Configuring a Siebel Janna VBC

To display Siebel Janna specific data within your Siebel application, you must configure a Siebel Janna VBC. A Siebel Janna VBC is a custom virtual business component that is configured from within Siebel Tools. Siebel Janna VBCs are based on Siebel Janna Data Objects. For every type of Siebel Janna Data Object that will flow into your Siebel application, a separate Siebel Janna VBC should be configured. The Siebel Janna VBC is based on the CSSJBCVExtern class.

Perform the following tasks to create and configure a Siebel Janna VBC:

- Collect information about the Siebel Janna Data and Siebel Janna Service Objects.
- Create the Virtual Business Component definition in Siebel Tools.
- Create the Virtual Business Component field definitions in Siebel Tools.
- Configure the Virtual Business Component's user properties.
- Compile the Virtual Business Component.

The following procedures demonstrate how to gather information required for the VBC, as well as how to create and configure the VBC in Siebel Tools.

To gather information required for the Siebel Janna VBC

1 Determine the ProgID of the Siebel Janna Data Objects.

- For core Siebel Janna Data Objects, consult the Siebel Janna Tools Library for the ProgID of the Siebel Janna Data Object being integrated.

  Most core Siebel Janna Data Object ProgIDs are formed from the database table name that forms the basis of the Siebel Janna Data Object, and take the following form:

  ```
  JDBOBJ.<database table or view name>.40
  ```

  For example, the ProgID of the Siebel Janna Data Object that represents a record in the ConPhone table in a Siebel Janna database will be:

  ```
  JDBOBJ.ConPhone.40
  ```

- For extension Siebel Janna Data Objects, consult the technical design documents of the extension implementation for the ProgIDs for Siebel Janna Data Objects that were part of the extension module.

- For client extension Siebel Janna Data Objects, the client technical design documents should contain the ProgIDs for Siebel Janna Data Objects that were part of the client implementation.

  Most custom Siebel Janna Data Object ProgIDs that are formed from extension tables or views by using the wizards will be in the following format:

  ```
  <Project Name>.<database table or view name>.1
  ```

  For example, if you created a Siebel Janna Data Object based on an extension table called Transactions_Data, and the project name specified in the wizard was TransactionsDO, the ProgID would be:

```
TransactionsDO.Transactions_Data.1
```

NOTE: If the Siebel Janna Data Object in question was not created using conventional Siebel Janna SDK wizards, then consult technical documentation to determine its ProgID.

2 Determine the ProgID of the Siebel Janna Service Object.

- For core Siebel Janna Service Objects, consult the Siebel Janna Tools Library for the Siebel Janna Service Object being integrated.

  Most Siebel Janna Service Object ProgIDs are formed from the SQL Table Name that the Siebel Janna Service Object maintains, and take the following form:

  ```
  JCESO.<database table or view name>
  ```

  For example, the ProgID of the Siebel Janna Service Object that represents a record set in the ConPhone table in a Siebel Janna database will be:

  ```
  JCESO.ConPhone
  ```

- For extension Siebel Janna Service Objects, consult the technical design documents of the extension implementation for Siebel Janna Service Objects that were part of the extension module.

- For client extension Siebel Janna Service Objects, the client technical design documents should contain the ProgIDs for Siebel Janna Service Objects that were part of the Client Implementation.

- Most custom Siebel Janna Service Object ProgIDs that are formed from extension tables or views by using the wizards will be in the following format:

  ```
  <Project Name>.<database table or view name>.1
  ```

  For example if you created a Siebel Janna Service Object based on an extension table called Transactions_Data, and the project name specified in the wizard was TransactionsSO, the ProgID would be:

```
TransactionsSO.Transactions_Data.1
```

NOTE: If the Siebel Janna Service Object in question was not created using conventional Siebel Janna SDK wizards, then consult technical documentation to determine its ProgID.

---

3. Determine the Siebel Janna Data Object Field Names. They will have properties that represent each column of the underlying SQL Table or View that the Siebel Janna Data Object is responsible for maintaining. However, the SQL Table or View Definition is not reliable as an indicator of the fields available from a Siebel Janna Data Object. Instead, the relevant Technical Documentation for the Siebel Janna Data Object, the OLEView type library display, or the Implementation Source Code should be consulted for the fields exposed by the Siebel Janna Data Object.

4. Determine the Siebel Janna Data Object Key Field Names by doing one of the following:

NOTE: Some of the properties of the Siebel Janna Data Object will be Key Fields. These act as foreign keys to other tables or as a primary key for the table the Siebel Janna Data Object maintains.

- Consult the Siebel Janna Tools, the extension technical documentation, or the client extension technical documentation for the names of the key fields on each Siebel Janna Data Object.
- Examine the database table or view for foreign key and uniqueness constraints to determine which fields are key fields. The columns attributed with these kinds of rules are key fields.
- Examine the Siebel Janna Data Object’s corresponding Siebel Janna Service Object source code, and the client source code for the Siebel Janna service object and Siebel Janna Data Object pair.
- Check for GUID values in the Siebel Janna Data Object’s properties. Siebel Janna Data Object properties that contain GUID values represent primary keys and foreign keys on a Siebel Janna Data Object.

5. Determine the Siebel Janna Data Object XML tag name. The XML tag name for a Siebel Janna Data Object generated by the Siebel Janna Tools will be identical to the name of the corresponding database object. If the Siebel Janna Data Object was based on an extension table called Transactions_Data, for example, the XML tag name would be Transactions_Data. Should the XML tag name not conform to this convention, you will need to consult technical documentation.

6. Determine the Siebel Janna Requestor ProgID. The Siebel Janna Requestor ProgID can be one of four values depending on the transport being used to activate Siebel Janna Service Objects on the SJAS.

   - For DCOM: JGenObj.JGenDCOMReq
   - For HTTP: JGenObj.JGenHTTPReq
   - For MSMQ: JGenObj.JGenMSMQReq
   - For IBMMQ: JGenObj.JGenIBMMQReq

7. Set a page threshold (optional). You can restrict the number of rows the VBC retrieves at each invocation using the Page Threshold value. In this way, VBCs that issue queries which return many records will only retrieve the first X (where X is the Page Threshold) records until the next page of records is requested by the application. This can improve VBC performance.

To create the Siebel Janna VBC in Siebel Tools

1. Locate the Siebel Janna Interoperability VBC Template Business Component in the Business Components Object Types in Siebel Tools.
2. Right-click the record on its row indicator and click Copy.
3. Enter a name for the new VBC.
4. Select the correct Project for your VBC.

To create the Siebel Janna VBC Fields in Siebel Tools

1. With the new Business Component Row Selected, browse to the Field Child Object Type in Siebel Tools.
2. Create a new Field Object Type Row for each Field in the Business Component.

3 Name the field according to naming conventions in *Siebel Tools Reference*, and use the default Field Object Type attributes.

To configure the Siebel Janna VBC's Field Mapping User Properties

1 The VBC's *Field Mapping* User Properties specify the mapping between a Siebel Janna Data Object and a Siebel Business Component. The format for these User Properties is as follows:

```
<Siebel Janna Data Object Field1>=<Siebel Business Component
Field1>;<Siebel Janna Data Object Field2>=<Siebel Business
Component Field2>...<Siebel Janna Data Object Fieldn>=<Siebel
Business Component Fieldn>
```

2 In cases where the Siebel Janna Data Object Field (JDOField) does not have a corresponding Siebel Business Component Field (BCField), then there should be no right side of the `<JDOField>=<BCField>` expression, reducing it to `<JDOField>=`

3 In cases where the Siebel Business Component Field (BCField) does not have a corresponding Siebel Janna Data Object Field (JDOField) then there should be no left side of the `<JDOField>=<BCField>` expression reducing it to `=<BCField>`

4 Where the Field Mapping property spans more than the allowable room on a Business Component User Property, it can be extended to an additional User Property called Field Mapping n where n is the total number of Field Mapping user properties used so far. See Step 1.

To configure the Service Parameters User Property

The *Service Parameters* User Property specifies information used by the Siebel Janna Gateway Service to communicate with the Siebel Janna Application Server. It consists of the following information gathered from prior steps:

- JSOProgID
- JDOProgID
- JDOTagName
- RecInfo
- SecInfo

- KeyField(s)

  It also consists of two additional optional pieces of information:

  - Criteria
  - UpdateType

  This information is entered into the service parameters user property, Step 1, in the following format:

  ```
  SOProgID=<value>;JDOProgID=<value>;JDOTagName=<value>;RecInfo
  =<true/false>;SecInfo=<true/
  false>;KeyField=<value>;UpdateType=<optional
  value>;Criteria=<optional value>
  ```

To compile the Siebel Janna business component

- From within Siebel Tools, compile the individual object or compile the project of which the Business Component is part. See *Siebel Tools Reference* for further information on this procedure.

Integrating the User Interface

A second potential approach is to implement User Interface (UI) Integration. It is similar to Data Integration in that the data comes from a Siebel Janna database using a distributed environment. Unlike the Data Integration approach, however, the Siebel Applet can be associated with either a VBC or a standard Siebel Business Component. At the presentation layer, the Applet acts as a host to the corresponding ASP page for the Siebel Janna zone. In effect, the Applet serves as a container for the ASP page, so the User Interface looks essentially as it does in the Siebel Janna application. For more information about VBCs and the Siebel Janna Gateway Business Service, see "Integrating Data" on page 26.

NOTE: Before you begin configuring your Siebel application for UI integration with Siebel Janna, please make sure your environment is configured correctly with a functioning Siebel Janna Web Client application and Siebel eBusiness application.

To configure Siebel for UI Integration with Siebel Janna, you must perform the following tasks:

- Identify the Siebel Janna Web Client Zone that will be hosted in Siebel.
- Create the applet component definition in Siebel Tools.
- Set the applet user properties in Siebel Tools.
- Add the applet to the appropriate view.
- Compile the applet and view.

To configure Siebel for integration with Janna

1. Identify the Siebel Janna Web Client Zone.

    a. Determine the Siebel Janna Web Client ASP Path for a particular Siebel Janna Zone by referring to the development documentation. You may also check the Siebel Janna Web Client JASPLink table and other associated configuration tables. The Siebel Janna Web Client Zone ASP path is found in the JASPLink table in the ASPFile column.

   - **b** Determine the ASP parameters required by the Siebel Janna Zone you wish to display in a Siebel View. The ASP parameter values are dependent upon individual Zone implementation. The best source for this information is the technical design and development documentation of the Zone. If this is unavailable, then the implementation should be reviewed by engineering or other technical resources to determine the correct ASP Parameter values.

2 Create the applet component definition in Siebel Tools.

   - **a** Locate the Siebel Janna Zone Adapter Applet in the Applet Object Types in Siebel Tools.
   - **b** Make a copy of this record by right-clicking on its row indicator and clicking Copy.
   - **c** Enter an appropriate name for the new applet row.
   - **d** Select the appropriate Project for your applet component.

3 Set the applet user properties in Siebel Tools.

   - **a** Using the information from Step 1, configure the ZoneSRC property of the new applet. Configure the Property as follows:

```
http:\\<server>\webview\ie\<zone ASP file Name>
```

   - **b** Using the information from Step 1, configure the ASPParams User Property with a semicolon-delimited list of the Fields from the Business Components whose values will be passed to the Request variables collection of the ASP Zone. The User Property has the following format:

```
<BusComp Field1>;<BusComp Field2>;...;<BusComp Fieldn>
```

4 Add the applet to the appropriate view using Siebel Tools, just as you would any other Siebel Applet.

5 Compile the applet and view.

For a list of definitions of the terms mentioned in the preceding sections, please see Appendix , "Siebel Janna Terminology and Architecture."

Siebel Janna Adapter Requirements

A Siebel Janna adapter would be a predefined business service that would function similar to the SAP BAPI Adapter. This adapter can be used as part of a workflow to send and receive information to Siebel Janna applications. The following are the requirements for enabling this business service.

- The business service should be a prebuilt business service similar to other eAI adapters like the SAP BAPI adapter.
- The business service should have one method: Execute.
- The method arguments for this business service should be as follows:
  - Integration Object Instance (the external integration object corresponding to the Siebel Janna service object)
  - Connect String
  - Username
  - Password
  - Output Integration Object (in case a response is received)

The Siebel Janna Service Objects in Siebel Janna applications can be represented as external integration objects. The requirement for the Integration Object Wizard to support Siebel Janna service objects is as follows:

- The Integration Object Wizard needs to support an additional object type called the Siebel Janna Service Objects.
- In Siebel Janna Tools this wizard will have the ability to read the metadata from Siebel Janna on these Siebel Janna Service Objects and create an external integration object corresponding to the Siebel Janna Service Objects.
- The object must work with both the 2000 and 2001 suites.

Siebel Janna Gateway Requirements

A Siebel Janna Gateway Business Service would interact with VBCs in Siebel and the Siebel Janna Service Objects in Siebel Janna to exchange information between the two systems. A Siebel Janna Gateway is similar to the XML gateway that currently exists.

To get to the Siebel Janna data, the Gateway Business Service calls the appropriate Siebel Janna Service Object. The Siebel Janna Service Object is responsible for performing actions on behalf of the client (Siebel). When Siebel requests that data be fetched from the Siebel Janna application, the results of the fetch come back in the form of a Siebel Janna Data Object.

The Siebel Janna Gateway Business Service must support the following methods:

- Delete
- Insert
- Init
- PreInsert
- Query
- Update

This business service takes the following information as parameters:

```
Service Object name
```

The parameter is specified when configuring the VBC and part of the Business Component user property. The format should be as follows:

```
Service Name = Siebel Janna Gateway

Service Parameters  = "ServiceObjectName=<Siebel Janna Service
Object Name>"
```

Interoperability Case Study 3

This chapter provides a case study in which to follow the configuration and implementation details of Chapter 2, “Interoperability,” using a fictional client, Capital Street Holdings (CASH). The case study addresses two objectives of Siebel Janna interoperability:

- Data Integration—Use Siebel eBusiness Application Integration (eAI) component methods to display and access data from an existing external Siebel Janna application.
- User Interface Integration—Display Siebel Janna User Interfaces within the Siebel application using an existing Siebel Janna framework.

The case study implementation is based on a scenario that involves an existing Siebel Janna custom solution as specified in the Siebel Janna Tools Tutorial. For more information regarding the technical aspects of the configuration, see the Tutorial section of the Siebel Janna Tools Library.

Working knowledge of Siebel Janna Tools, Siebel Tools, and Siebel eBusiness Application Integration is beneficial to comprehend this case study. Prior to reading this chapter, it is also recommended you review information and procedures from Chapter 2, “Interoperability.”

Capital Street Holdings—Introduction

Capital Street Holdings Inc. (CASH) is a firm of Financial Advisors, who integrated their legacy portfolio management system with their Customer Relationship Management information to access it in a contact-centric fashion. Siebel Janna Contact Enterprise was customized to provide the necessary information.

Existing Implementation

The existing implementation of Siebel Janna Contact Enterprise has the following custom components:

The Transaction Zone

Displays a tabular listing of trading activity by account for the selected contact. Each trade is a separate line item. The table will:

- Display Account, Transaction Date, Transaction Type, Ticker, Quantity, Open Price, and Commission columns.
- Allow columns to be resized and moved.
- Be able to sort by each column.
- Be able to filter the trades by Transaction Type.
- Be able to selectively display only those transactions that match the Ticker in the user’s selection in the Summary Zone.

The Summary Zone

Displays a tabular listing of trade summaries by Ticker for the selected Contact. Each trade summary will be a separate line item. The table will:

- Display Ticker, Description, Quantity, Avg. Cost, Total Cost, Market Price, Market Value, and Gain/Loss columns.
- Allow columns to be resized and moved.
- Be able to sort by each column.

- Allow the selection of any row so that related transactions will be displayed in the Transaction Details Zone.

The Quick Search Capability

The Quick Filter was enhanced to include a custom tab that displays two fields allowing users to:

- Select Contacts who have traded a specified Ticker.
- Select Contacts who have shares in excess of a specified volume.

Field Validation

Custom code was written to validate the Country field in the Contact Detail form so if the entry in the Country field is:

- C or CAN or Can, it is automatically converted to Canada upon insert or update.
- USA, it is automatically converted to U.S.A. upon insert or update.

Interoperability Objectives

The objective for CASH is the portability of core Siebel Janna zones into the Siebel application, which allows for interoperability between the two products and the use of Siebel Janna data within the Siebel product. Any Siebel Janna custom extensions developed for CASH are not considered in this case study.

References

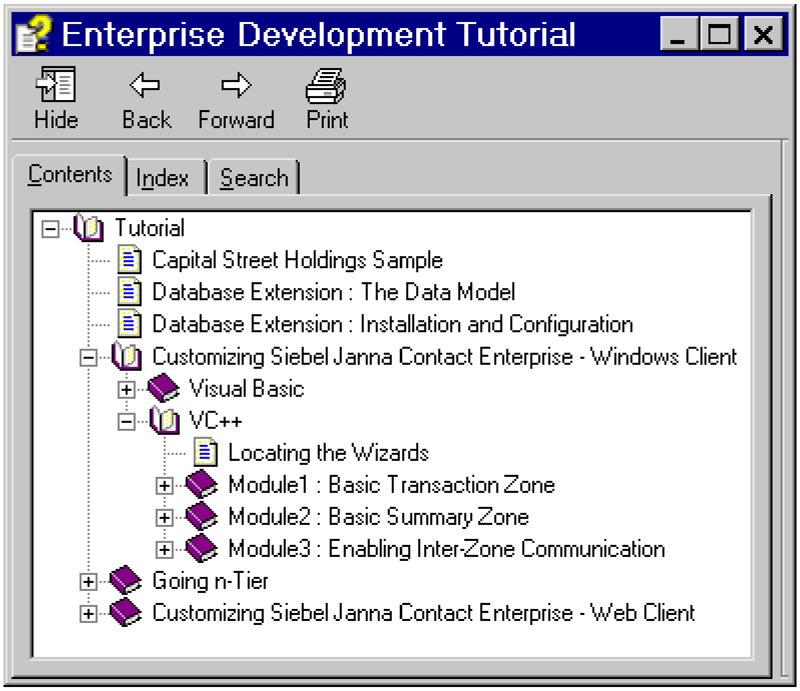
The case study implementation is based on a scenario that involves an existing Siebel Janna custom solution as specified in the Siebel Janna Tools Tutorial. To access the tutorial, you must have the Siebel Janna Tools development environment installed. Running the Tutorial.chm file launches a help file that contains detailed instructions on the configuration of Siebel Janna Contact Enterprise.

Transaction Zone—Implementation Details

The implementation objective for this case study is the creation of a list-type zone in the Siebel Janna Institutional Finance Web Client and then porting this zone into a Siebel 7 Financial Services application, to create an interoperable application. Perform the following steps under each heading to achieve this interoperable state between applications.

Locating the Siebel Janna Wizards

Locate the Siebel Janna Wizards by following the steps outlined in the Siebel Janna Tutorial.chm file. Select Tutorial > Customizing Siebel Janna Contact Enterprise—Windows Client > VC++ > Locating the Wizards. See Figure 2 for details.



Tutorial
- Capital Street Holdings Sample
- Database Extension : The Data Model
- Database Extension : Installation and Configuration
- Customizing Siebel Janna Contact Enterprise - Windows Client
  - Visual Basic
  - VC++
    - Locating the Wizards
    - Module1 : Basic Transaction Zone
    - Module2 : Basic Summary Zone
    - Module3 : Enabling Inter-Zone Communication
- Going n-Tier
- Customizing Siebel Janna Contact Enterprise - Web Client

Figure 2. Siebel Janna Applications Development Tutorial—Tutorial.chm file

After following the preceding steps, when your Visual C++ studio is launched, the following wizards are visible: Siebel Janna Data Object Wizard, Siebel Janna Service Object Wizard, and Siebel Janna Presentation Object Wizard.

Creating the Siebel Janna Data Object

Create the Siebel Janna Data Object by following the steps outlined in the Siebel Janna Tutorial.chm file.

Select Tutorial > Customizing Siebel Janna Contact Enterprise—Windows Client > VC++ > Module 1: Basic Transaction Zone > Creating the Transactions Data Object.

After following the preceding steps, the TransactionsDO.dll is created.

Creating the Siebel Janna Service Object

Create the Siebel Janna Service Object by following the steps outlined in the Siebel Janna Tutorial.chm file.

Select Tutorial > Customizing Siebel Janna Contact Enterprise—Windows Client > VC++ > Module 1: Basic Transaction Zone > Creating The Transactions Service Object.

After following the preceding steps, the TransactionsSO.dll is created.

Creating the Siebel Janna Presentation Object

Create the Siebel Janna Presentation Object by following the steps outlined in the Siebel Janna Tutorial.chm file.

Select Tutorial > Customizing Siebel Janna Contact Enterprise—Windows Client > VC++ > Module 1: Basic Transaction Zone > Creating The Transactions Presentation Object.

After following the preceding steps, the TransactionsPO.dll is created, as well as the TransactionsWorker.asp and TransactionsMain.asp.

Configuring the Transaction List Zone

Configure the Transaction List zone by following the steps outlined in the Siebel Janna Tutorial.chm file.

Select Tutorial > Customizing Siebel Janna Contact Enterprise—Windows Client > VC++ > Module 1: Basic Transaction Zone > Inserting the Presentation Object into Janna Contact.

After following the preceding steps (excluding step 10 and 11), the Transactions List zone is configured in the Contacts > Details tab.

After performing steps 10 and 11, the Transactions List zone is now visible in the Web Client (http://localhost/webview).

Determining the ProgID of Siebel Janna Data Object

Depending on the kind of Siebel Janna Data Object (Core Siebel Janna Data Object, Extension Siebel Janna Data Object, Client Extension Siebel Janna Data Object) there are several ways to determine the correct ProgID.

- For core Siebel Janna Data Objects, consulting the Siebel Janna Tools documentation should provide the correct ProgID for the Siebel Janna Data Object being integrated.
- For extension Siebel Janna Data Objects, consulting the Technical Design Documents of the extension implementation should contain the ProgIDs for Siebel Janna Data Objects that were part of the extension module.
- For client extension Siebel Janna Data Objects, the Client Technical Design Documents should contain the ProgIDs for Siebel Janna Data Objects that were part of the client implementation.

In the absence of accurate documentation, there are a number of tools that will allow you to determine ProgID based on the compiled DLL (regedit or OLEView) that implements a DLL. If a compiled DLL is not available, use the source code of the Siebel Janna Data Object Implementation to determine its ProgID. Most often Siebel Janna Data Object ProgIDs are formed from the SQL Table Name that the Siebel Janna Data Object maintains and take the following form:

```
<Module Name>.<SQL Table or View Name>.1
```

where <Module Name> is often <SQL Table or View Name> with DO appended to it. For example, a Siebel Janna Data Object that maintains the SQL View vSomeViewData would have a ProgID of vSomeViewDataDO.vSomeViewData.1.

Determining the ProgID of Siebel Janna Service Object

Depending on the kind of Siebel Janna Service Object (Core Service Object, Extension Service Object, Client Extension Service Object) there are several ways to determine the correct ProgID.

- For core Siebel Janna Service Objects, consulting the Siebel Janna Tools documentation should provide the correct ProgID for the Siebel Janna Data Object being integrated.
- For extension Siebel Janna Service Objects, consulting the technical design documents of the Extension implementation should contain the ProgIDs for Service Objects that were part of the Extension module.
- For client extension Siebel Janna Service Objects, the client technical design documents should contain the ProgIDs for Siebel Janna Service Objects that were part of the Client Implementation.

In the absence of accurate documentation, there are a number of tools that will allow a user to determine ProgID based on the compiled DLL (regedit or OLEView) that implements a DLL. If a compiled DLL is not available, use the source code of the Siebel Janna Service Object Implementation to determine its ProgID. Most often, Siebel Janna Service Object ProgIDs are formed from the SQL Table Name that the Siebel Janna Service Object maintains and take the following form:

```
<Module Name>.<SQL Table or View Name>.1
```

where <Module Name> is often <SQL Table or View Name> with SO appended to it. For Example, a Siebel Janna Service Object that maintains the SQL View vSomeViewData would have a ProgID of vSomeViewDataSO.vSomeViewData.1.

Determining the Siebel Janna Data Object Field Names

The Siebel Janna Data Object has properties that represent each column of the underlying SQL table or View that the Siebel Janna Data Object is responsible for maintaining. However, the SQL table or View Definition is not reliable as an indicator of the fields available from a Siebel Janna Data Object. Instead, the relevant technical documentation for the Siebel Janna Data Object, the OLEView type library display, or the implementation source code should be consulted for the fields exposed by the Siebel Janna Data Object.

The following field names must be present: Account, Commission, ContactGUID, EditUserGUID, GUID, OpenPrice, RecInfo, Quantity, SecInfo, Ticker, TransactionData, TransactionType, and ViewUserGUID.

Determining the Siebel Janna Data Object Key Field Names

Some of the properties of the Siebel Janna Data Object will be Key Fields that act as foreign keys to other tables or as a primary key for the table the Siebel Janna Data Object maintains. There are a few ways to determine which properties of the Siebel Janna Data Object represent these key fields. Consult the Siebel Janna Tools, the extension technical documentation, or client extension technical documentation for the names of the key fields on each Siebel Janna Data Object.

In the absence of the appropriate technical documentation, there are still methods to determine which fields are key fields. The first, and most accessible, is to examine the database table or view for foreign key constraints and uniqueness constraints. The columns attributed with these kinds of rules are key fields.

Without access to the database schema, the next reliable method to determine the key fields on a Siebel Janna Data Object is by examining its corresponding Siebel Janna Service Object source code and the client source code for the Siebel Janna Service Object and Siebel Janna Data Object pair.

Finally, if the source is not available, and the database schema cannot be examined, then most often Siebel Janna Data Object properties that contain GUID values represent primary keys and foreign keys on a Siebel Janna Data Object.

The following field names must be present: ContactGUID (FK—UserProf), EditUserGUID (FK—UserProf), ViewUserGUID (FK—UserProf), GUID (PK), and Ticker (FK—Ticker Data).

Determining the Siebel Janna Data Object XML Tag Name

In cases where the Siebel Janna Data Objects have been developed with the help of Siebel Janna Wizards in Visual C++, the XML tag name will be equivalent to the database object name. In this case, for example, where the Siebel Janna Data Object is driven by a database table named Transactions_Data, its XML tag name will also be Transactions_Data. Technical documentation needs to be reviewed should this assumption not yield the correct XML tag name. As a last resort, the user can determine the Siebel Janna Data Object XML tag name by using published help files for the Siebel Janna Tools to create a shell program using core classes (that is, use ConvertDO2XML method for JcomHelper class). Support for developing this interface is found by searching for ConvertXML2DO in the Index tab of the SDK.chm help file, which ships with Siebel Janna Tools.

Determining the Janna Requester ProgID

The Janna Requestor ProgID is one of four values depending on the transport being used to activate Siebel Janna Service Objects on the Siebel Janna Application Server:

- For DCOM: JGenObj.JGenDCOMReq
- For HTTP: JGenObj.JGenHTTPReq
- For MSMQ: JGenObj.JGenMSMQReq
- For IBMMQ: JGenObj.JGenIBMMQReq

Creating the Siebel Janna Virtual Business Components

Follow the procedure below to create a Siebel Janna Virtual Business Component in Siebel Tools. Refer to *Siebel Tools Reference* for detailed information on this process.

To create a Siebel Janna Virtual Business Component

1. Start Siebel Tools.
2. Lock the Siebel Janna Interoperability project.
3. Locate the Siebel Janna Interoperability VBC Template Business Component in the Siebel Janna Interoperability project.
4. Copy this record and give the new business component an appropriate name.
5. Choose an appropriate project for the new business component.

The Siebel Janna Virtual Business Component: Transaction List Project: Siebel Janna Interoperability is the result of this procedure.

Creating Field Definitions for the Siebel Janna VBC

Follow the procedure below to create field definitions for the Siebel Janna Virtual Business Component in Siebel Tools. Refer to *Siebel Tools Reference* for detailed information on this process.

To create field definitions for the Siebel Janna VBC

1. Start Siebel Tools.
2. Lock the Siebel Janna Interoperability project.
3. Highlight the new business component created previously. Locate the Field Child Object Type.
4. For each field in the Siebel Janna Data Object, create a new field object type record with appropriate names and attributes.

Configure the Siebel Janna VBC's User Properties

Perform the two following procedures to configure the Siebel Janna Virtual Business Component in Siebel Tools. Refer to *Siebel Tools Reference* for detailed information on this process.

To configure field mapping user properties

Using the information garnered previously, configure Field Mapping user properties. These User Properties specify the mapping between a Siebel Janna Data Object and a Siebel Business Component. The format for these User Properties is:

```
<Janna Data Object Field1>=<Siebel Business Component
Field1>;<Janna Data Object Field2>=<Siebel Business Component
Field2>…<Janna Data Object Fieldn>=<Siebel Business Component
Fieldn>
```

In cases where the Siebel Janna Data Object Field (JDOField) does not have a corresponding Siebel Business Component Field (BCField) then there should be no Right Hand Side of the <JDOField> = <BCField> expression, reducing it to <JDOField> =.

In cases where the Siebel Business Component Field (BCField) does not have a corresponding Siebel Janna Data Object Field (JDOField) then there should be no Left Hand Side of the <JDOField> = <BCField> expression reducing it to = <BCField>.

Where the Field Mapping property spans more than the allowable room on a Business Component User Property, it can be extended to an additional user property called Field Mapping n where n is the total number of Field Mapping user properties used so far.

To configure the Service Parameters User Property

The service parameters User Property specifies information used by the Janna Gateway Service to communicate with the Siebel Janna Application Server. It consists of the following information gathered in the previous procedure:

- SOProgID
- JDOProgID
- JDOTagName

- RecInfo
- SecInfo
- KeyField(s)

As well as two additional optional pieces of information:

- Criteria
- UpdateType

This information is entered into the Service Parameters user property in the following format:

```
SOProgID=<value>;JDOProgID=<value>;JDOTagName=<value>;RecInfo=<t
rue/false>;SecInfo=<true/
false>;KeyField=<value>;UpdateType=<optional
value>;Criteria=<optional value>
```

Compile the Siebel Janna Virtual Business Component and Project

Perform the necessary procedures to compile the business component in Siebel Tools. Refer to *Siebel Tools Reference* for detailed information on this process.

Create Applet and View

Perform the necessary procedures to create an applet and appropriate view in Siebel Tools based on he Siebel Janna Virtual Business Component. Refer to *Siebel Tools Reference* for detailed information on this process.

Compile SRF file

Perform the necessary procedures to compile the SRF file in Siebel Tools. Refer to *Siebel Tools Reference* for detailed information on this process.

Launch Siebel Financial Services 7.0

Launch the Siebel Financial Service 7.0 application and verify that the Siebel Janna zone is visible within the Siebel Financial Services application.

Capital Street Holdings—Other Functionality

After performing the procedures from the preceding section, the Siebel Janna Zone is now visible in the Siebel Financial Services application.

Test the product thoroughly to make sure the Siebel Janna functionality operates as intended in the Siebel Financial Services application.

The following sections briefly explain the other Janna Contact Enterprise Zones from CASH and how they are implemented into an interoperable application. See "Existing Implementation" on page 40 for further information.

The Summary Zone

The summary zone is created in the same way as the Transactions zone in the preceding example.

The Quick Search Capability

Siebel Janna provided quick filters on specific data sets, and these required the development of custom Siebel Janna Query Objects. The functionality, however, is provided in the standard Siebel 7 application through the querying capability provided in List and Form Applets. Therefore, there is no need to replicate any of the Siebel Janna Tools Query Object technical steps into the Siebel application.

Field Validation

Siebel Janna allowed field validation for controls on the User Interface through a class called the Siebel Janna Running Object. These were essential checks executed on the client side to affect the entry of data into controls. In a Siebel application, therefore, such functionality can be implemented through client-side scripting using Siebel eScript.

Siebel Janna Data Objects A

This appendix contains a list of Siebel Janna Data Objects and their specifications.

ActionTemplate

ActionTemplate	Specifications
DO ProgID	JDBOBJ.ActionTemplate.40
SO ProgID	JCESO.ActionTemplate
Field Names	ActionTemplateGUID Alarm AlarmLead AlarmLeadUnits AlarmObjGUID AlarmType BitmapID Desc ForContact ForUser FullRead Keyword Memo ObjectGUID PPGUIDn Priority ProgID Prop RecInfo RetainInterval RetainNum

ActionTemplate	**Specifications**
Field Names *(continued)*	RetainType Scripts Selected ShowIn ShowObjectIcon StatusUser GUID
Key Field	ActionTemplateGUID
XML Tag Name	ActionTemplate

AddressType

AddressType	Specifications
DO ProgID	JDBOBJ.AddressType.40
SO ProgID	JCESO.AddrType
Field Names	AddressTypeGUID AutoAddress Comments Desc Default Properties RecInfo
Key Field	AddressTypeGUID
XML Tag Name	AddressType

ConAddress

ConAddress	Specifications
DO ProgID	JDBOBJ.ConAddress.40
SO ProgID	JCESO.Addresses
Field Names	Address1 Address2 Address3 AddrGUID AutoAddr City ConGUID Country Default Desc Properties RecInfo State Zip
Key Field	AddrGUID
XML Tag Name	ConAddress

ConDocConGroup

ConDocConGroup	Specifications
DO ProgID	JDBOBJ.ConDocConGroup.40
SO ProgID	JCESO.Doc
Field Names	Group GUID Status
Key Field	GUIDGroupGUID
XML Tag Name	ConDocConGroup

ConDocRecur

ConDocRecur	Specifications
DO ProgID	JDBOBJ.ConDocRecur.40
SO ProgID	JCESO.ConDocRecur
Field Names	DocGUID Month RecurGUID Rinterval RSDate Rtype WeekDay Year
Key Field	RecurGUID
XML Tag Name	ConDocRecur

ConDocUserGroup

ConDocUserGroup	Specifications
DO ProgID	JDBOBJ.ConDocUserGroup.40
SO ProgID	JCESO.ConDocUserGroup
Field Names	Alarm AlarmDate AlarmLead AlarmLeadUnits AlarmObjGUID AlarmType DocGUID Prop SchPlusID Sequence UserGUID UserStatus
Key Field	DocGUIDUserGUID
XML Tag Name	ConDocUserGroup

ConIntCode

ConIntCode	Specifications
DO ProgID	JDBOBJ.ConIntCode.40
SO ProgID	JCESO.ConIntCode
Field Names	ConGUID Desc IntCodeGUID RecInfo
Key Field	ConGUIDIntCodeGUID
XML Tag Name	ConIntCode

ConPhone

ConPhone	Specifications
DO ProgID	JDBOBJ.ConPhone.40
SO ProgID	JCESO.ConPhone
Field Names	ConGUID Default Desc IDXNumber Number NumberType PhoneGUID Properties RecInfo
Key Field	ConGUIDPhoneGUID
XML Tag Name	ConPhone

Contact

Contact	Specifications
DO ProgID	JDBOBJ.Contact.40
SO ProgID	JCESO.Contacts
Field Names	Addresses
	Affix
	BaseData1
	BaseData2
	BaseData3
	Branch
	Company
	ConGUID
	ConType
	ConTypeGUID
	CustomFields
	Dear
	Deleted
	EntityType
	FirstName
	Fullname
	FullRead
	Initials
	IntCodes
	LastName
	Level
	ParentGUID
	PhoneNumbers
	Position
	RecInfo

Contact	Specifications
Field Names *(continued)*	SecInfo Targeted Title
Key Field	ConGUID
XML Tag Name	Contact

Contact Document

Contact Document	Specifications
DO ProgID	JDBOBJ.ContactDocument.40
SO ProgID	JCESO.Doc
Field Names	ActionTemplateGUID BitmapID ConDocConGroups ConDocRecur ConDocUserGroups ConGUID ContactList ContactName EDate Desc DocGUID Dur FolderGUID FullRead IsAlarmSet IsFolder Keyword Memo MultipleContacts MultipleUser ObjectGUID OleProgID Parent Priority ProjGUID

Contact Document	**Specifications**
Field Names *(continued)*	Prop RecInfo RecurGUID RetainDate RetainInterval RetainNum RetainType SecInfo SDate SPlusID ShowIn ShowObjectIcon Status StatusDesc UserGUID UserList
Key Field	DocGUID
XML Tag Name	ContactDocument

ConType

ConType	Specifications
DO ProgID	JDBOBJ.ConType.40
SO ProgID	JCESO.ConTypes
Field Names	Comments ConTypeGUID DeptLiteral Desc EntityType IconID Properties RecInfo SubFlag
Key Field	ConTypeGUID
XML Tag Name	ConType

ConUDF

ConUFD	Specifications
DO ProgID	JDBOBJ.ConUDF.40
SO ProgID	JCESO.UDFValue
Field Names	ConGUID ConGUIDValue CurrencyValue Date Literal NumberValue Properties RecInfo UDFGUID UDFValueGUID Value ValueType
Key Field	UDFValueGUID
XML Tag Name	ConUdf

Group

Group	Specifications
DO ProgID	JDBOBJ.Group.40
SO ProgID	JCESO.Groups
Field Names	GroupName GroupGUID
Key Field	GroupGUID
XML Tag Name	Group

GroupSecurity

GroupSecurity	Specifications
DO ProgID	JDBOBJ.GroupSecurity.40
SO ProgID	NA
Field Names	EditGroup EditGroupName EditSecurity EditSecurityDesc ViewGroup ViewGroupName ViewSecurity
Key Field	NA
XML Tag Name	SecInfo

IntCode

IntCode	Specifications
DO ProgID	JDBOBJ.IntCode.40
SO ProgID	JCESO.IntCode
Field Names	Comments Desc IntCodeGUID Properties RecInfo
Key Field	InCodeGUID
XML Tag Name	IntCode

JSISync

JSISync	Specifications
DO ProgID	JDBOBJ.JSISync.40
SO ProgID	JCESO.JSISync
Field Names	ConduitID ConduitType ConGUID DocGUID ObjectType Prop SyncGUID UserGUID
Key Field	SyncGUID
XML Tag Name	JSISync

OleObject

OleObject	Specifications
DO ProgID	JDBOBJ.OleObject.40
SO ProgID	JCESO.Ole
Field Names	
Key Field	
XML Tag Name	JSIOleObject

Phone

Phone	Specifications
DO ProgID	JDBOBJ.Phone.40
SO ProgID	JCESO.Phone
Field Names	Class IsOleObjectDirty IStorage ObjGUID RecInfo Version
Key Field	
XML Tag Name	Phone

Pref

Pref	Specifications
DO ProgID	JDBOBJ.Pref.40
SO ProgID	JCESO.Pref
Field Names	Comments Default Desc NumberType PhoneTypeGUID Properties RecInfo
Key Field	
XML Tag Name	Pref

QuickPick

QuickPick	Specifications
DO ProgID	JDBOBJ.QuickPick.40
SO ProgID	JCESO.QuickPick
Field Names	Comments Desc Field RecInfo
Key Field	
XML Tag Name	QuickPick

RecordInfo

RecordInfo	Specifications
DO ProgID	JDBOBJ.RecordInfo.40
SO ProgID	NA
Field Names	RecCreateDate RecCreateUID RecSecLvl RecSyncDate RecSyncUID RecUpdateDate RecUpdateUID
Key Field	NA
XML Tag Name	RecInfo

Script

Script	Specifications
DO ProgID	JDBOBJ.Script.40
SO ProgID	JCESO.Script
Field Names	ActiontemplateGUID Script Trigger Version
Key Field	
XML Tag Name	Script

SecFunction

SceFunction	Specifications
DO ProgID	JDBOBJ.SecFunction.40
SO ProgID	JCESO.Functions
Field Names	FuncDesc Function ModuleDesc Object ObjectDesc RecInfo SecLevel
Key Field	
XML Tag Name	SecInfo

SPlusSync

SPlusSync	Specifications
DO ProgID	JDBOBJ.SplusSync.40
SO ProgID	JCESO.SPlusSync
Field Names	ChangeNumber ScheduleID
Key Field	
XML Tag Name	SPlusSync

UDF

UDF	Specifications
DO ProgID	JDBOBJ.UDF.40
SO ProgID	JCESO.UDFDef
Field Names	Comments Desc Properties RecInfo Sort TabGUID Type UDFGroup UDFGUID
Key Field	
XML Tag Name	Udf

User

User	Specifications
DO ProgID	JDBOBJ.User.40
SO ProgID	JCESO.User
Field Names	ActionSecDefaults
	Company
	ContactSecDefaults
	CurrTab
	DataPath
	Department
	DocPath
	DocumentSecDefaults
	EmailAddr
	EmailId
	Fax
	FirstLoginDate
	LastLoginDate
	Name
	NotificationAddress
	NTUserID
	NumberOfRemoteLogins
	NumberOfServerLogins
	Password
	Position
	Phone
	PrimaryGroup
	RecInfo
	ReportsToGUID
	SecLevel

User	Specifications
Field Names *(continued)*	SortBy UserGUID UserGroupsCollection UserGroupString UserId
Key Field	
XML Tag Name	User

UserGroup

UserGroup	Specifications
DO ProgID	JDBOBJ.UserGroup.40
SO ProgID	JCESO.UserGroups
Field Names	GroupGUID GroupName UserGUID UserId UserName
Key Field	
XML Tag Name	UserGroup

VersionInfo

VersionInfo	Specifications
DO ProgID	JDBOBJ.VersionInfo.40
SO ProgID	JCESO.Version
Field Names	glb_id_db glb_id_rec glb_id_sys Version
Key Field	
XML Tag Name	VersionInfo

Siebel Janna Terminology and Architecture B

This appendix contains partial background information on Siebel Janna terminology and its object architecture. For information on Siebel application architecture, consult the *Siebel Bookshelf*.

Siebel Janna Terminology

Table 1 describes terms for the Siebel Janna Upgrade. For equivalent Siebel terminology, please consult *Glossary* on the *Siebel Bookshelf*.

Table 1. Siebel Janna Terminology

Term	Definition
Binary Large Objects (BLOB)	Large objects such as pictures, word processing documents, spreadsheets, and so on.
Siebel Janna Data Object	Siebel Janna Data Objects transport the state of Siebel Janna Service Objects throughout the architecture. They represent the most basic component and typically encapsulate a single record in a table or a view from a database.
Presentation Object	Communicates with Requestors to collect information from the user interface (UI) or to present information to the UI.
Requestors	Clients communicate with the Siebel Janna Application Server through Requestors. There are Requestors for each transport protocol, such as DCOM, MQ Series, MSMQ, and so on. Requestors provide transparent connection to the Siebel Janna Service Objects, which can be local and remote.
Siebel Janna Service Object	A Siebel Janna Service Object is a COM component that retrieves and processes data from a database. In a typical n-tier architecture, these objects form the business layer where business rules are enforced. Normally, Siebel Janna Service Objects run under the Siebel Janna Application Server process.
Siebel Janna Database Access Objects (SJDAO)	Siebel Janna Database Access Objects are a layered hierarchy of COM objects used to manipulate Siebel Janna data.

Siebel Janna Object Architecture

Siebel Janna applications are based on open software architecture for the cross-platform development of n-tier applications. It is extended using the object-oriented technology known as the Component Object Model (COM).

Siebel Janna Applications are extended because of its COM-based architecture. Each logical layer is composed of COM-based components that you can reuse, customize, extend, or integrate with other applications and objects. The Siebel Janna Tools online help provides documentation and samples to help you use this extensible architecture to meet your business needs.

Architecture Layers

Siebel Janna Applications are divided into three logical layers:

- Presentation layer
- Business layer
- Data layer

Each layer is composed of objects. A standard implementation of a Siebel Janna Application uses core Siebel Janna objects to provide a standard Customer Relationship Management solution. Because enterprise requirements are diverse, Siebel Janna Applications allow you to customize objects at each layer and integrate them into Siebel Janna Applications.

The Presentation Layer

The presentation layer allows the user interface (UI) to display or collect data. This layer is also called the front-end layer, or the workflow or presentation logic, of the application. The presentation layer can be Windows-based or Web-based and contains presentation objects which interact with the end user. Presentation objects can be created from ActiveX controls, Active Server Pages, or Java Applets, depending upon whether the platform is Windows-based or Web-based. Siebel Janna Applications use ActiveX controls for presentation objects for the Windows Client and Active Server Pages (with JavaScript and HTML) for presentation objects for the Web Client.

Presentation objects communicate with the rest of the system through requestors. Requestors are COM objects which call other objects in the business layer to perform tasks. They are important because they provide a transparent connection to objects in the business layer, objects which could be local or remote.

The Business Layer

The business layer, also referred to as the application layer, is composed of the following:

- Siebel Janna Service Objects
- Siebel Janna Listener Objects
- Siebel Janna Application Server (SJAS), a component manager
- Janna Database Access Objects (SJDAOs)

On a two-tier system, the listener object and the SJAS are not required because object management is done locally by COM. Two-tiered applications usually work very well in departmental-scale applications with modest numbers of users (under 100), a single database, and secure, fast networking. On a three-tier system, the Requestors and listener objects isolate the presentation and business layers from the transport protocol. The independence between the business layer, the presentation layer, and the data layer provides a number of benefits, as follows:

- Developers can use powerful development tools such as Visual Basic and Visual C++ to develop reusable application components, instead of using more limited procedural languages.
- Administrators can replicate application components to run on multiple machines simultaneously. This configuration spreads client loads across multiple machines. Application component replication (as opposed to data replication) is not possible with two-tiered architectures because stored procedures must run in a single database.
- Application components can share database connections. This lowers the number of total sessions that the database server must support. With two-tiered systems, the database must allocate a connection for every user.

- Middle-tiered application components can be secured centrally using a common infrastructure. Administrators can grant or deny access on a component-by-component basis.
- Access to resources such as mainframe applications and other databases comes through native protocols and application interfaces instead of through data gateways. This allows application owners to control access to their data.

Siebel Janna Service Objects

A Siebel Janna Service Object is an ActiveX component that retrieves and processes data from a database. In a typical n-tier architecture, these objects form the business layer where business rules are enforced.

In the Siebel Janna architecture, the presentation objects communicate their data requirements at a more abstract level and do not hard code details of how or where data is physically stored, fetched, and managed. Also, they do not perform low-level data manipulation. Siebel Janna Service Objects perform these tasks.

Siebel Janna Applications provide one Siebel Janna Service Object for each of the most commonly used tables in the Siebel Janna database. Because Siebel Janna Applications' open architecture allows for easy extensibility, it is common to have your own database schema integrated with the Siebel Janna database. The data in these extension tables need to be accessed and processed to supply the often complex data requirements of the custom presentation objects. Custom Siebel Janna Service Objects authored by the developer will manage the storage, retrieval, and processing of data from these extension tables.

NOTE: Siebel Janna Service Objects need not only access data from a database. You can customize Siebel Janna Service Objects to enforce other business rules or manage or use other Siebel Janna Service Objects through aliasing.

Siebel Janna Application Server (SJAS)

SJAS provides a server-centric environment for developing and deploying three-tiered Siebel Janna Customer Relationship Management solutions based on COM. It is a key component of Siebel Janna Applications. With SJAS, Siebel Janna Service Objects run under the control of SJAS on servers and are invoked by presentation objects (through Requestors) running on clients through Microsoft's Distributed COM (DCOM) or Microsoft Message Queue Server (MSMQ) technologies, or MAPI. Clients can be a mixture of conventional applications and Active Server Page (ASP) scripts running within Microsoft Internet Information Server (IIS).

Siebel Janna Data Access Objects (SJDAOs)

Siebel Janna Data Access Objects (SJDAOs) are a layered hierarchy of COM objects used to manipulate Siebel Janna data. The SJDAO is comprised of two layers—the physical layer and the logical layer. Each object has one or more properties that define its characteristics and one or more methods that you use to perform various operations on the object.

- **The Physical Layer.** This physical layer wraps the functionality of the ODBC API into a series of COM objects that physically connect and manipulate data in an ODBC database. The objects are maintained in the JDUODBC.dll library. If the database goes offline, the physical layer insulates the application from abnormal termination and reconnects to the database when it is online again.

- **The Logical Layer.** The logical layer is the programming interface for the Siebel Janna Database Engine or the physical layer. Objects represent each table in the Siebel Janna database. Each object has a set of properties that define its characteristics and one or more methods that you use to perform various operations on the object.

Siebel Janna Data Objects

Siebel Janna Data Objects transport the state of business objects throughout the architecture. They represent the most basic component of the Siebel Janna architecture and typically encapsulate a single record in a table or view from a database.

The Data Layer

The data layer represents the myriad of data sources distributed across an enterprise system. These sources could be Siebel Janna databases or any other data sources. Data sources containing a Siebel Janna application database can be accessed through ODBC. Data on other data sources can be accessed using any industry data access technologies like ActiveX Data Objects.

Object Interaction Between Layers

In response to a user interaction, when a presentation object requires information, it will request that information through the requestor in the presentation layer. The requestor routes the request to the appropriate Siebel Janna Service Object. You can have as many Siebel Janna Service Objects as you wish. The requestor is unique because it does not need to know where a Siebel Janna Service Object resides, just the name of the Siebel Janna Service Object. The requestor will read the registry, which tells it the location and communication method required to access the Siebel Janna Service Object that will fulfill the request from a presentation object.

In the business layer, SJAS runs as an NT Service. Once started, SJAS establishes a pool of database connections to support the multi-tasking of database interactions. Next, the Siebel Janna Application Server starts a series of listener threads that wait for a request to the database from the Presentation Layer. When a request is received, SJAS creates an instance of the appropriate business object designed to handle the request. Once the request is passed to the business object, SJAS returns to its state of listening or processes the next request. SJAS can handle multiple requests simultaneously when deployed in a multiprocessor environment.

The Siebel Janna Service Objects do some intelligent work. If the service consists of a data request, a JDAO thread is captured (or it will wait until a thread is free) and the data request is sent to the database through the JDAO. The database may return a stream of data. This stream is broken into individual packets of information and sent to the Siebel Janna Service Object. These packets are called Siebel Janna Data Objects and usually represent a single record from a table or view in the database. The Siebel Janna Data Objects are returned to the presentation objects that sent the request.