Siebel Web UI Dynamic Developer Kit Guide

Version 8.0, Rev. B

March 2010

ORACLE

Contents

1 What's New in This Release

What's New in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0, Rev. B

Table 1 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 1. New Product Features in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0, Rev. B

Topic	Description
"About the LOV Service" on page 53	Added a footnote to the Filter Value integration object component field stating that you cannot use this field for searching.

Additional Changes

The following changes were also made for Version 8.0, Rev. B:

- Oracle WebLogic replaces BEA WebLogic.
- Oracle Workshop for WebLogic replaces BEA WebLogic Workshop.
- Java EE replaces J2EE.

What's New in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0, Rev. A

Table 2 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 2. New Product Features in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0, Rev. A

Topic	Description
Client-specified session types See "About Authentication and Session Management" on page 11 and *Integration Platform Technologies: Siebel Enterprise Application Integration*.	Added the ServerDetermine session type. Before v8.0, only None, Stateful, and Stateless were supported.
Dynamic picklists without pick applets See "Dynamic Picklists Without Pick Applets" on page 20 and "Example of How the DDK Wizard Creates Dynamic Picklists Without a Pick Applet" on page 29.	Added two new topics that explain how the Web UI DDK Wizard creates dynamic picklists for views in the user interface when a business component field does not have a pick applet defined. Before v8.0, picklists without defined pick applets were not supported.
Strongly typed integration object See *Siebel Business Process Framework: Workflow Guide* and *Siebel Object Types Reference*.	There is a new data type for defining workflow process properties specifically used by Web services. Strongly typed integration objects allow you more functional scripts and better performance.

What's New in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0

Table 3 lists changes described in this version of the documentation to support release 8.0 of the software.

Table 3. New Product Features in Siebel Web UI Dynamic Developer Kit Guide, Version 8.0

Topic	Description
State Management Type business service property field See "About Authentication and Session Management" on page 11 and *Integration Platform Technologies: Siebel Enterprise Application Integration*.	A new feature for Siebel Order Management—Web Channel session state management—requires that all business services are marked with a setting describing the Object Manager client-session mode when making a request to the Siebel Server. Valid states are: Stateless, Stateful, and Server Managed.
EAI file streaming enhancement See *Integration Platform Technologies: Siebel Enterprise Application Integration*.	Streaming of EAI incoming requests and outgoing responses is now supported.

2 Web UI DDK Overview

This chapter includes the following topics:

About the Web UI DDK

Many companies rely on service-oriented architectures to present data and functionality from multiple applications in a single user interface, such as a portal application. The Web UI Dynamic Developer's Kit (DDK) solution helps companies expose Oracle's Siebel data and functionality in portal applications and custom UIs. The Web UI DDK solution is based on Web services and consists of interfaces that allow access to Siebel data and a wizard for generating a quick-start kit. The quick-start kit includes sample JavaServer pages that can be deployed in a Web application and that provide Web developers with sample code demonstrating common data manipulation operations and basic UI rendering. Web developers can modify the sample code instead of developing from scratch, giving them a jump-start on development work.

Key features of the Web UI DDK solution are as follows:

- **Web UI DDK Wizard.** This wizard in Siebel Tools is used to generate the development kit that includes sample JavaServer pages and other Java EE artifacts.

  For more information, see "About the Web UI DDK Wizard" on page 13.

- **Sample JavaServer Pages.** These are sample pages generated by the Web UI DDK Wizard and then deployment in a Java EE environment.

  For more information, see "About the Sample JavaServer Pages" on page 18.

- **Data Schema and Metadata Report.** This is a report that Java developers use to understand the Siebel data model.

  For more information, see "Generating the Data Schema Report" on page 26.

- **UI Data Adapter.** This Siebel business service exposes APIs for accessing Siebel data.

  See also "About the UI Data Adapter and UI Data Sync Services" on page 10.

  For detailed information about the UI Data Adapter, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

Web UI DDK Architecture Overview

Figure 1 illustrates the components of the Web UI DDK solution architecture. The diagram shows Web UI DDK Wizard in Siebel Tools and run-time components on the Java EE Web application server, the Web server, and the Siebel Server. It also shows the components involved in an external authentication scenario (LDAP).

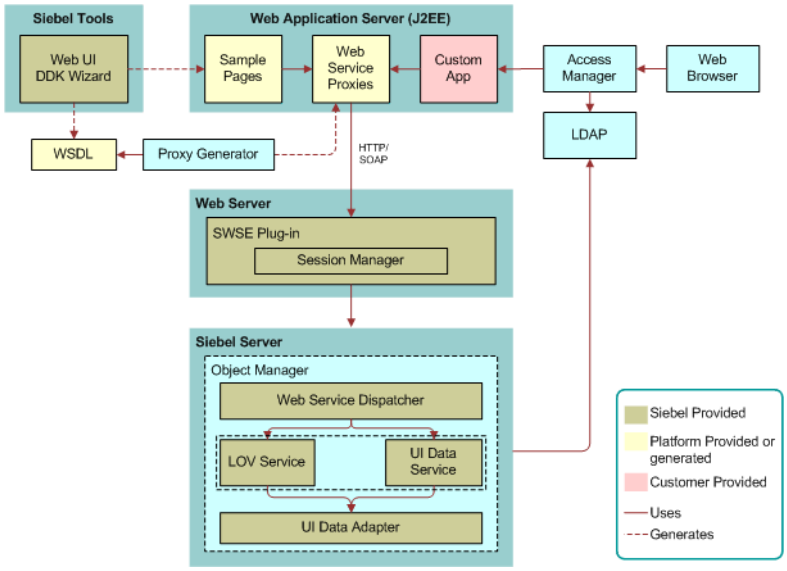


Figure 1. Web UI DDK Architecture Overview

The Web UI DDK Wizard guides the Siebel application developer in creating the object definitions in the Siebel object repository and generates WSDL files, sample JSP Pages, and other Java artifacts that can be used by the Java Application Developer to render the custom user interface.

The sample JSP pages are based on the Siebel Web service infrastructure. Web services define the data that request and response messages exchange, the data processing requirements, and the data mapping to and from the underlying software applications. The Siebel Web service infrastructure receives standard XML messages from the Java EE Web application server environment and then invokes a business service or business process (workflow) to process the message. After processing requests, the Siebel Server returns a response message to the Java EE Web application server.

The Web UI DDK Wizard generates WSDL files for a given Siebel view. Java client proxies generated from the WSDL files contain APIs that a custom UI can use to set and get authentication and session information in SOAP headers. After a successful login, session and authentication information is passed in a session token included in the SOAP header. The session token is an encrypted version of the session ID and user credentials. The SWSE is responsible for parsing the inbound soap header to retrieve the session token and user credentials.

Web Service Dispatcher is a service that processes SOAP messages for Siebel Web services. It is responsible for parsing the SOAP envelope on request, generating the correct SOAP envelope for response, and formatting SOAP faults for response. It is also responsible for dispatching the XML data in the SOAP body to an appropriate business service or business process (workflow) as specified in the Inbound Web Service Administration view.

The UI Data Adapter and UI Data Sync services support managing Siebel data. Web services based on these business services expose a strongly typed interface that can be used for data manipulation and data retrieval operations.

In single sign-on implementations, the Access manager and LDAP server are responsible for authenticating requests and then forwarding them to the Java EE server.

The following bullets summarize the components shown in Figure 1:

- **Web UI DDK Wizard.** This wizard generates repository objects, Web services administrative data, WSDL (Web Service Definition Language) files, and sample JSP pages.

  For more information, see "About the Web UI DDK Wizard" on page 13.

- **Proxy Generator.** This component generates Java client proxies based on WSDL files produced by the DDK Wizard. The readme.txt file in the DDK Wizard's output folder has instructions for using the proxy generator.

  For more information about the readme.txt file, see "Using the Readme File to Deploy Sample JavaServer Pages" on page 34.

- **Sample Pages**. The DDK Wizard generates sample JavaServer pages that the Java developer can deploy and run in a custom-built Web application.

  For more information, see "About the Sample JavaServer Pages" on page 18.

- **Custom Web Applications.** These are customer-built Web applications using technologies from Oracle and IBM, such as Oracle Workshop for WebLogic or IBM WebSphere Studio Application Developer. These applications host JavaServer pages and perform tasks such as:

  - Preparing queries and making service calls.
  - Passing session tokens in requests and extracting fresh tokens from responses.
  - Handling exceptions coming from Web service proxy calls.
  - Handling retry logic when there are problems connecting to the Web server.

- **Web Application Server (Java EE).** This server hosts sample pages and custom Web applications.

- **Session Management.** A Siebel-supplied component on the Siebel Web Server Extension (SWSE) that handles session information and authentication information.

  For more information, see "About Authentication and Session Management" on page 11.

- **Access Manager.** Access Manager authenticates user credentials with LDAP, sets the security token in the request, and then forwards the token to the Java EE server.

  For more information, see "About Authentication and Session Management" on page 11

- **Object Manager.** The object manager provides the session environment in which the application runs, processes requests, and interacts with the following:

  - **Web Service Dispatcher.** A business service that processes SOAP messages for Siebel Web services. The Web Service Dispatcher is responsible for parsing the SOAP envelope on request, generating the correct SOAP envelope for response, and formatting SOAP faults for response. It also dispatches the XML data in the SOAP body to an appropriate business service or business process for processing.
  - **UI Data Adapter.** This adapter provides data access to business objects and business components, and exposes interfaces for performing operations such as insert, delete, update, and query.
  - **UI Data Sync Services.** These are business services, based on the UI Data Adapter class, that provide access to a given business object, such as account. UI Data Services are made available to custom-built Web applications by exposing them as Web services. You can either use preconfigured UI Data services or create new UI Data services.
  - **LOV Service.** When given an List of Values (LOV) name and language code, this service provides access to LOVs.
  - **Session Access Service**. This business service exposes simple operations, such as Ping and Echo, that can be used to start a session. It also exposes operations that can access profile attributes associated with a session.

    For more information about Web Service Dispatcher and UI Data Adapter, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

    For more information about the UI Data Sync, LOV, and Session Access services, see Chapter 6, "About Preconfigured Web Services."

About the UI Data Adapter and UI Data Sync Services

The UI Data Adapter business service enables custom-built Web applications to access Siebel functionality and data. Business services based on the UI Data Adapter class is called a UI Data Sync Service. These business services can be published as Web services. They provide the following:

- Access to business components and business objects.
- Strongly-typed data for appropriate rendering of the user interface.
- Strongly-typed APIs that enable access to operations such as insert, delete, update, query, and execute.
- Support for various query modes, including query by example.
- Support for downloading and uploading of file attachments.

For more information about the UI Data Adapter, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

For more information about UI Data Sync Services, see "About UI Data Sync Services" on page 51.

About Authentication and Session Management

The Siebel Web UI DDK supports Siebel Authentication and Session Management SOAP headers, including Web single-sign authentication. These security features are part of the Siebel Web services framework and are summarized below:

- **Siebel Authentication and Session Management SOAP Headers.** Custom Web applications invoke Siebel sessions by sending Web service requests to the Siebel Web Server Extension (SWSE). Requests include Siebel Authentication and Session Management SOAP headers that can include user credentials for logging in and session information for reconnecting to established sessions.

  SOAP headers enable you to pass a session token back and forth between the custom Web application and the Siebel Server. The session token supports Stateless, Stateful, and Server Determine sessions. The session token is encrypted and consists of a session ID and user credentials. The Session Manager on the SWSE extracts a session ID and user credentials from the session token and then reconnects to an open login session on the Siebel Server. In the case of a server failure, the Session Manager uses the user credentials to create a new login session. After the request is processed, a new session token is passed back in the outbound response SOAP header.

  NOTE: The sample JSP pages generated by the Web UI DDK Wizard use Apache Axis. For information about custom coding needed to implement Siebel session management and authentication SOAP headers using tool sets native to IBM WebSphere and Oracle WebLogic, see Chapter 5, "Using Native Web Service Technology Stacks."

- **Web Single-Sign On.** Siebel Web services support Web single sign-on deployment scenarios in which third-party applications handle authentication and then pass authentication information to the Siebel application. Once authenticated by the third-party application, users do not have to explicitly log in to the Siebel application.

For more detailed information about session type, session management, and Siebel Web services and security, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

For more information about general security topics, see *Siebel Security Guide*.

About Developer Roles for Using the Web UI DDK

Tasks for generating and deploying the Web UI DDK are divided between Siebel developers and Java developers.

Siebel Developers

Siebel developers typically perform these tasks:

- Analyzing Siebel data models and object models
- Running the DDK Wizard for a given view to generate JavaServer pages and Java EE artifacts
- Verifying Integration Objects and Business Services in Siebel Tools
- Verifying Web service administration data
- Compiling the SRF with repository changes
- Handing off the Java artifacts generated by the Web UI DDK Wizard to Java Web developers

Java Web Developers

Java developers typically perform these tasks:

- Importing Java artifacts into a Java development environment
- Generating Java proxy objects from WSDL
- Deploying sample pages as working examples
- Using the metadata and schema report to understand the Siebel data model
- Writing custom code to invoke Siebel logic and manage data in the Siebel database using proxy objects

Scenario for Using the DDK Wizard

This section gives one example of how you could use the Web UI DDK Wizard. You can use the Web UI DDK Wizard differently, depending on your business model.

A company has a business requirement to enable partners to create and manage Siebel accounts using a custom-built Web application.

The company's Siebel developer reviews the high-level requirements and determines that the All Account List View is the best view to expose to partners in the custom-built Web application. The Siebel developer opens the Web UI DDK Wizard and selects the appropriate integration object and Web service to use to expose the view. The Siebel developer completes the wizard, generating a developer kit that includes sample JavaServer pages for the view and all the supporting Java EE artifacts required to deploy the view in a Java EE application. The Siebel developer archives the files and directs the Java developer to the file location.

The Java developer uses the readme.txt file in the DDK output folder to deploy the JavaServer pages. The Java developer imports the Java EE artifacts and libraries into the Java development environment and then uses Apache Ant (Java-based build tool) to generate Java proxy objects from the generated WSDL files.

As a proof-of-concept, the Java developer shows the sample JavaServer pages to the business owners and demonstrates how to use the pages to manage accounts. The Java developer uses the sample JavaServer pages as working examples and the Data Schema report to understand the object hierarchy and object metadata. Using the sample JSP pages as a starting point, the Java developer develops custom code to expose the All Account List view in the Web application.

About Data Access Control

You can administer access control by associating responsibilities to business services and business service methods. This lets you configure the application so that users who access a Siebel application from the custom-built Web application user interface have different levels of access to Siebel data.

For details about administering access control for business services, see *Siebel Security Guide*.

About the Web UI DDK Wizard

The Web UI DDK Wizard enables the Siebel developer to generate a development kit with sample JavaServer pages and other Java EE artifacts that can be passed on to the Java developer and deployed in a custom-built Web application. The sample JavaServer pages are based on a Siebel view and provide Java developers with working examples of how to use Web services to expose Siebel functionality in a custom-built Web application.

The Web UI DDK Wizard guides the Siebel developer through the process of associating Web services with views, MVGs, and picklists. It provides a link to the Data Access Service Wizard, which enables the Siebel developer to configure Web services, including creating integration objects and business services. The end result is a development kit that includes everything that the Java developer needs to run the sample JavaServer pages in a custom-built application.

Siebel developers select a given Siebel view in Siebel Tools, and then run the Web UI DDK Wizard. The wizard walks the developer through the process of mapping data from the view to an integration object and creating integration objects and UI Data Sync business services. The Wizard automatically publishes business services as Web services and generates the sample JavaServer pages and Java EE artifacts that represent the selected view and can be deployed in a Web application. Figure 2 summarizes the process of using the Web UI DDK Wizard.

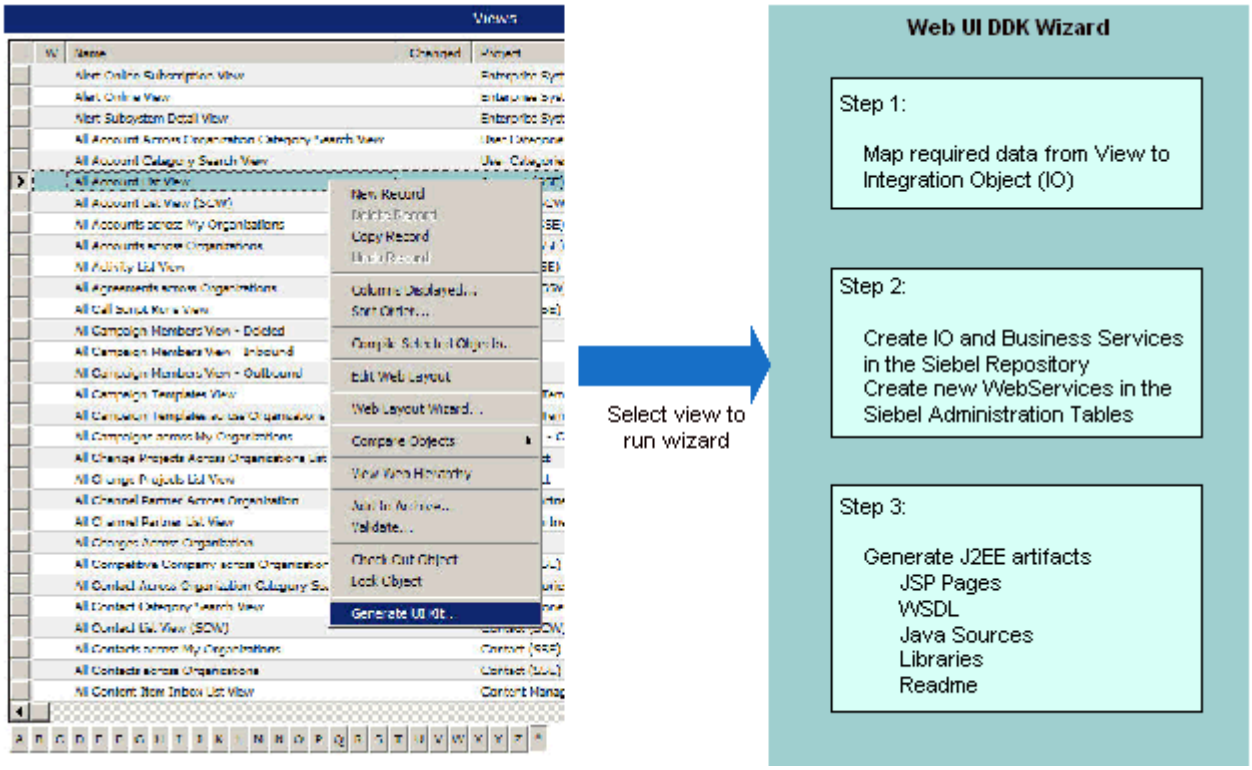


Figure 2. Web UI DDK Wizard

For information about running the Web UI DDK Wizard, see "Running the Web UI DDK Wizard" on page 25.

The remaining sections in this topic cover details about each screen in the Web UI DDK Wizard.

- "The Repository and Administrative Data Changes Screen" on page 17
- "The Web UI DDK Options Screen" on page 18

NOTE: The screens marked with asterisks are integrated into the Web UI DDK Wizard from the Data Access Service Wizard. You can also access the Data Access Service Wizard in Siebel Tools by choosing File, New Object, and then choosing the EAI tab.

The Web UI Dynamic Developer Kit Wizard Screen

Use this screen to select the applets to display on the sample JavaServer pages. The Web UI DDK supports list and form applets only.

- Parent applets are applets whose business component is the same as the primary business component associated with the view's business object.
- Child applets are applets whose business component has a parent-child relationship with the primary business component associated with the view's business object.

The Web Service Data Source Screen

Use this screen to associate a view and any MVG fields or dynamic picklist fields with Web services. You must associate each MVG and picklist field you want to edit with a Web service. After you click Next, the wizard validates the integration object associated with the Web service and synchronizes it with the selected component.

NOTE: The synchronization process makes changes to repository objects that require you to recompile the SRF.

If no Web service is available, you can click Create Web Service to open the Data Access Service Wizard and create a new Web service. If you click Create Web Service, the "The Integration Object Page" appears.

Web services are available to associate to views, MVGs, and picklists as described in Table 4.

Table 4. Summary of Web Service Reuse

Object Type	Web Services Are Available for a Given View, MVG, or Picklist When...
View	■ The Web service port refers to a business service (UI Data Sync Service). ■ The business service (UI Data Sync Service) refers to an integration object with an External Name Property that matches the name of the view's business object. The integration object is referenced in the Integration Object property of the Business Service Method Arguments. ■ The integration object has a root integration component with an External Name that matches the Primary Business Component of the view's business object. ■ The integration object has a Base Object Type property set to *Siebel Business Object*.
MVG or Picklist	■ The Web service refers to a business service (UI Data Sync Service). ■ The business service (UI Data Sync Service) refers to an integration object that has a root integration component with an External Name property that matches the primary business component of the dynamic picklist or MVG applet. ■ The Base Object Type property of the integration object is set to *Siebel Business Object*.

The Integration Object Page

Use this page to indicate whether you want base your Web service on an existing integration object or create a new one. Typically, you need an integration object for each business object you want to expose.

You must select a project to associate with the integration object. Only locked projects appear in Project drop-down list.

The Integration Object Builder Page

Use this page to enter the source object for a new integration object.

- If you are creating a Web service for a view, the source object is the name of the business object associated with the view.
- If you are creating a Web service for an MVG or a picklist, the available values in the source object drop-down list are business objects containing a primary business component that can represent the data for the picklist or the MVG.

- If you open the Data Access Wizard from the Web UI DDK Wizard, the root component is automatically populated. If you open the Data Access Wizard as a stand-alone wizard, you must select the root component.

CAUTION: Do not include spaces in the integration object name. During deployment, spaces in the names of integration objects cause spaces to appear in the package names generated by Axis. The Java compiler does not accept spaces in package names.

The Integration Object Builder-Choose Integration Components Page

Use this page to select the integration components to be activated.

NOTE: You can deselect the entire tree of integration components by clicking the check mark from the top-level object. This removes the check mark from the top-level object and from all child components in the tree.

The Web UI DDK Wizard validates and synchronizes the integration object. The synchronization process automatically selects any integration components or integration component fields that are needed to render the view, even those you have not selected in this page of the Data Access Service Wizard.

The UDS Methods Page

Use this page to see the methods included in the UI Data Sync business service. All available methods are selected by default.

CAUTION: all methods must be selected, otherwise the Web UI DDK files will not compile.

The Web Service Page

Use this page to define a URL for the Web service. This information is stored in the Address field of the Web service port in the Administration Web service.

The Repository and Administrative Data Changes Screen

This screen displays the following information for you to review:

- Errors that prevent the Web UI DDK Wizard from proceeding. If errors occur, Next is disabled and you must go back to previous screens to correct the problems.
- Repository changes to be made. After reviewing the changes, click Next to commit the changes to the database.
- A list of dynamic picklists or MVGs fields that have not been associated with a Web service.

The Web UI DDK Options Screen

Use this screen to specify the location where the sample JSP pages and other Java artifacts generated by the wizard will be stored.

About the Files Generated by the Web UI DDK Wizard

The Web UI DDK Wizard guides developers through the process of creating object definitions in the Siebel repository and defining Siebel Web services. The Web UI DDK Wizard generates the following files that are used by the Java developer to render a custom UI in a Web application. The generated files are saved in the DDK output folder specified in the Web UI DDK Wizard.

- **Sample JavaServer Pages.** These are pages that render the UI for the selected view. JavaServer pages are also created for Login and any dynamic picklists (that is, picklists that do not use LOVs) and MVG fields that have been associated with Web services. For more information, see "About the Sample JavaServer Pages" on page 18.

  NOTE: Sample JSP pages are based on Apache Axis and use JSP Standard Tag Libraries.

- **WSDL Files.** Web Service Definition Language files represent the business object and components for the selected view. WSDL files are also generated for any non-LOV picklists and MVG fields that have been associated with a Web service. The DDK output folder includes the Apache Ant build file for generating Java proxies from the WSDL files.
- **Java Files.** These utility Java source files perform common tasks such as login and session management.
- **Readme File.** This is a text file (readme.txt) that gives instructions on deploying the sample pages in a Java EE Web application.
- **DDK Log.** This is a log file (DDK.log) that captures errors and tracing information from the code generation process.
- **Cascading Style Sheets and Images**. The sample JavaServer pages use these.

About the Sample JavaServer Pages

The sample JavaServer pages that the Web UI DDK generates demonstrate the basics of using Siebel Web services to expose Siebel data and functionality in a custom-built Web application. The sample JSP pages are working examples for the Java developer and a starting point for development. The sample JavaServer pages are designed to be deployed for learning purposes only.

NOTE: The sample JSP pages are based on Apache Axis and use standard JSTL Tag Libraries.

The major features of the sample JavaServer pages are as follows:

- **Login Page.** This page has fields for entering user credentials and a button to invoke the login action. A login Web service invokes the login operation. The login operation takes the username and password as input parameters and returns a session token from the server. This session token is then stored in the Java EE session. For more information about session tokens and authentication, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

- **Session Token.** The session token returned from the login operation is stored in the SOAP header and sent with every Web service call.
- **CRUD Operations.** To support CRUD operations, list applets modes are List, Base, Edit, New, and Query. Form applet modes are Base, Edit, New, and Query. MVG and dynamic picklist applets modes are New and Query.

  NOTE: CRUD is an acronym for Create, Read, Update, and Delete. CRUD reports show how a process affects given data objects, specifically whether objects are Created, Read, Updated, or Deleted.
- **Record Set Navigation.** List applets and the base mode of form applets include a hyperlink that enables the user to navigate to the previous or next set of records.
- **MVG.** MVG and MVG association applets are created for MVG fields that are associated with Web services. The primary field is not displayed in the parent applet.
- **Dynamic Picklist.** A dynamic picklist applet is created for a picklist field associated with a Web service. The parent applet contains one picklist field that shows the combined values of the primary record.
- **LOV.** LOVs are displayed in a drop-down list. The LOV service is called to retrieve the values of the LOV. It takes the LOV Type as an input parameter.

The sample JavaServer pages demonstrate basic features for managing data in a view. The functionality characteristics of the sample JavaServer pages, including functionality not supported, are described in the following sections.

Lists of Values

Sample JavaServer pages do not support the following types of LOVs:

- Organization-enabled LOVs
- Hierarchical LOVs

Dynamic Picklists

Sample JavaServer pages do not support the following types of dynamic picklists:

- Unbound picklist
- Hierarchical picklist

Dynamic picklists or MVG fields in a dynamic pick applet are ignored. MVGs are treated as child objects and displayed in a child applet.

Dynamic Picklists Without Pick Applets

A dynamic picklist allows the user to populate a field in a view in the user interface by selecting values from a list that are exposed by way of a pick applet. The Siebel Web UI Dynamic Developer Kit (DDK) treats all dynamic picklists the same by creating a new JavaServer page for each business component field. However, when no pick applet is defined for a business component field, the DDK must create a JavaServer page with an imitation single-column picklist to show records from which the user can choose a value.

The process by which the picklist without pick apple creation occurs is similar to how Java artifacts are generated, but this creation is transparent to developers and users. You might find it useful, however to understand the logic behind this process. For more information about how the Web UI DDK creates dynamic picklists without pick applets, see "Example of How the DDK Wizard Creates Dynamic Picklists Without a Pick Applet" on page 29. For more information about generating Java artifacts, see "Generating Java Artifacts" on page 23.

List Applets

List applets in sample JavaServer pages have the following characteristics:

- List applets that do not have displayed columns will cause an error.
- The list column order is determined by examining the Edit List Applet Web Template and retrieving the item identifier.
- List Columns with the following characteristics are displayed in the list applet:
  - Available = Y
  - Show In List = Y
  - Inactive = N or Is Null
  - List column is not an MVG field
  - The corresponding business component field exists in the business component
- The Read Only property of the List Column object is ignored.

Form Applets

Form applets in sample JavaServer pages have the following characteristics:

- The generated JavaServer pages are based on the template defined in the Applet Mode property of the View Web Template Item.
- Controls with the following characteristics are displayed:
  - Mode = Null
  - Inactive = N or Is Null
  - Grid Property = FormattedHtml for applets in Grid Layout
  - Not an MVG
  - Corresponding business component field exists in the business component

MVG Applets

MVG applets in sample JavaServer pages have the following characteristics:

- Primary fields are not supported.
- The following properties of the multi-value link object are ignored:
  - No Associate
  - No Copy
  - No Delete
  - No Insert
  - No Update
- The layout of the MVG applet is based on the MVG Applet Web Template, if the fields exist in the source business component.
- Dynamic picklists or MVG fields defined in the MVG applet are ignored. MVGs are treated as child objects and displayed in a child applet. The primary field on the parent applet is not displayed.
- After inserting a new record, query, update, and delete are available in an MVG applet (1:M).

MVG Association Applets

MVG association applets in sample JavaServer pages have the following characteristics:

- Applet layout is based on the Applet Web Template of the MVG applet defined in the MVG applet property under the Applet Control.
- Dynamic picklists or MVG fields defined in the MVG association applet are ignored.

Drilldown Fields

Drilldown fields in sample JavaServer pages have the following characteristics:

- If an active drilldown object with a Name property set to Original is defined for an applet, the Hyperlink Property of the drilldown object acts as the drilldown field.
- If the drilldown object is not defined or the Hyperlink field is not visible in the applet, then the first visible field acts as the drilldown field.

CRUD Operations

Buttons such as New, Edit, and Delete are not always rendered on sample JavaServer pages, even if these operations are allowed. Execute operations that are not permitted will throw and exception.

Controlling CRUD operations using visibility rules, roles and responsibilities, configuration, and dynamic field values is not supported. For example, a field called Active in the Quote List Applet - NB determines whether or not an update operation is permitted. The sample JavaServer pages do not support this behavior.

User Interface

User interface functionality in sample JavaServer pages are summarized as follows:

- Data fields appear as text boxes, except LOV and Boolean fields.
- Predefault values are not supported.
- Previous and Next buttons are enabled, even on the beginning and last page.

Web Services

Web services that have the same namespace but different ports cannot be used together during the sample page generation for a given view.

About Preconfigured Web Services and Repository Objects

Several preconfigured Web services and their supporting repository objects (integration objects and business services) are delivered as seed data. They are available for you to reuse when running the Web UI DDK Wizard and the Data Access Service Wizard. However, they are set to inactive by default and must be activated and compiled before you can use them.

Web services and corresponding integration objects and business service exist for views associated with the following business objects:

- Account
- Contact
- Service Request
- Opportunity
- Orders

The names of these preconfigured Web services, business services, and integration objects begin with WC_ (WC followed by an underscore)—for example, *WC_Account_BS*. The name of the project in the Siebel repository that contains the preconfigured business services and integration objects is *Custom UI Samples*.

For more information about UI Data Sync Services, see “About UI Data Sync Services” on page 51.

3 Generating Java Artifacts

This chapter includes the following topics:

Process of Generating Java Artifacts

Perform the following tasks to generate a quick-start kit that contains sample JavaServer pages and the required Java EE artifacts needed to run the JavaServer pages in a Java EE Web application:

1. "Analyzing Siebel Views to Expose" on page 23
2. "Activating Preconfigured Integration Objects and Business Services" on page 24
3. "Activating Preconfigured Web Services" on page 24
4. "Running the Web UI DDK Wizard" on page 25
5. "Compiling the SRF" on page 25
6. "Generating the Data Schema Report" on page 26

Analyzing Siebel Views to Expose

Before running the Web UI DDK Wizard you need to identify the following:

- The view you want to expose in the custom-built Web application.
- The applets on the view to expose and verify that they are supported.

  Review "About the Web UI DDK Wizard" on page 13 and "About the Sample JavaServer Pages" on page 18.
- Any MVG or pick list fields to expose.

This task is a step in the "Process of Generating Java Artifacts" on page 23.

Activating Preconfigured Integration Objects and Business Services

Preconfigured integration objects and business services are inactive in the Siebel repository by default. Inactive objects—and any Web Services that refer to the inactive objects—will not appear in the Web UI DDK Wizard. Before you can use them, you must activate them and then recompile the SRF. See also "About Preconfigured Web Services and Repository Objects" on page 22.

This task is a step in the "Process of Generating Java Artifacts" on page 23.

NOTE: The LOVService and SessionAccess business services are active in the Siebel repository. However, the Web Services based on these business services are inactive. For information on activating Web Services, see "Activating Preconfigured Web Services" on page 24.

To activate integration objects and business services

1. Log on to Siebel Tools.
2. Activate integration objects as follows:
   a. In the Object Explorer, select the Integration Object Type.
   b. In the Object List Editor, query for integration objects that begin with WC_.
   c. Select the integration object you want to use, and then change the Inactive property to FALSE.
3. Activate business services as follows:
   a. In the Object Explorer, select the Business Service object type.
   b. In the Object List Editor, query for business services that begin with WC_.
   c. Select the business service that you want to use, and then change the Inactive property to FALSE.
4. Compile the changes to the SRF.

Activating Preconfigured Web Services

Web Services for common objects, such as Account, Contact, and Opportunity, are preconfigured and delivered as seed data. You must activate preconfigured Web Services before you can use them.

See also "About Preconfigured Web Services and Repository Objects" on page 22.

This task is a step in the "Process of Generating Java Artifacts" on page 23.

To activate preconfigured Web Services

1. Log on to the Siebel Web Client.
2. Navigate to the Administration - Web Services screen, then the Inbound Web Services view.

3 Query for Web Services that begin with WC_.

NOTE: Web Services preconfigured for use with the Web UI DDK begin with WC_.

4 Select the Web Service you want to use, and then change the Status field to *Active*.

5 Click the Clear Cache button for the changes to take effect.

The active Web Service becomes available as a choice in the Web UI DDK Wizard.

Running the Web UI DDK Wizard

You can run the Web UI DDK Wizard for active views when a business object has a primary business component associated with it.

NOTE: You cannot run the Web UI DDK Wizard for views not included in your current license key.

This task is a step in the "Process of Generating Java Artifacts" on page 23.

To run the Web UI DDK Wizard

1 In the Siebel Tools Object Explorer, select the View object type.

2 In the Object List Editor, select a view, right-click, and then choose Generate UI Kit.

The Web UI DDK Wizard opens.

For details of each screen in the wizard, see "About the Web UI DDK Wizard" on page 13.

3 After the wizard finishes, compile your changes to the SRF.

Sample JavaServer pages cannot run until the changes the Web UI DDK Wizard made to repository objects are compiled into the SRF and deployed on the server that will be handling requests from the Web application.

For an example of running the Web UI DDK Wizard, see "Example of Generating Java Artifacts" on page 26.

Compiling the SRF

When using the Web UI DDK solution, you need to compile the SRF in the following circumstances:

- After you have activated integration objects and business services. Inactive objects do not appear in the Web UI DDK Wizard.
- After you have run the Web UI DDK Wizard. The wizard either creates new objects or makes changes to existing objects during the synchronization and validation process.

This task is a step in the "Process of Generating Java Artifacts" on page 23.

Generating the Data Schema Report

Web application developers may not be familiar with the structure and relationships among business components and business objects in the Siebel data model. You can create a Data Schema report for a given integration object that summarizes this information in an easy-to-read format.

The Data Schema report includes the following:

- Hierarchical relationship between the business object and business component.
- Business component properties, such as SearchSpec and SortSpec, Table, No Insert, No Delete, No Merge.
- Field properties, such as Required, Data Type, and Read Only.

To generate the data schema report

1. In the Siebel Tools Object Explorer, select the Integration Object object type.
2. In the Object List Editor select an integration object.
3. Choose Reports, then Data Schema.

   The Data Schema report appears in the Siebel Report Viewer.

   You can print or save the report.

Example of Generating Java Artifacts

This topic gives one example of generating Java artifacts using the Web UI DDK Wizard. You may use this feature differently, depending on your business model.

This example assumes that Siebel Tools, Siebel Web Client, and the Siebel Server are installed and running.

To generate the Web UI DDK

1. Open Siebel Tools and log in using the following user credentials:
   - Username: SADMIN
   - Password: SADMIN
2. Create a new project called *MyAccountProject* and lock it.
3. In the Tools Object Explorer, choose the View object type.
4. Query for the All Account List View (SCW).
5. In the Object List Editor, select the view, right-click, and then choose Generate UI Kit.

   The first page of the Web UI Dynamic Developer Kit Wizard Opens. The Parent Applet field displays the applets that belong to the selected view:

   - Account List Applet (SCW)

- Account Form Applet (SCW)

6 On the Web UI Dynamic Developer Kit Wizard page, click Next.

The Web Service Data Source page appears, displaying the selected view in the left window.

You can expand the tree to display the applets on the view and any MVG or LOV fields on the applets. The window on the right displays Web Services that can be used to expose the applets, MVG fields, or LOV fields. These could be predefined Web Services that are shipped in the Siebel Repository (preconfigured Web Services begin with the prefix WC_) or Web Services that you have already created. If necessary, you can also create new Web Services.

7 Click Create New Web Service.

The Integration Object Page of the Data Access Service Wizard opens. You can use preconfigured integration objects or create a new one.

NOTE: Objects preconfigured for commonly used objects begin with the prefix WC_.

8 On the Integration Object Page, select *MyAccountProject*, select Create new Integration Object, and then click Next.

The Integration Object Builder page appears.

9 Enter a name for the new integration object, such as *TestAccountIO*, and then click Next.

NOTE: The name of the integration object must not include spaces.

The Integration Object Builder - Choose Integration Components page displays the available integration components for the selected integration object.

10 Accept the default selections, and then click Next.

The Business Service page appears.

11 Enter TestAccountBS for the business service name, accept the selected UI Data Service for the base service name, and then click Next.

The UDS Methods page appears.

12 On the UDS Methods page, make sure all methods are selected, and then click Next.

NOTE: All methods on the UDS Methods page must be selected for the DDK to compile.

The Web Service page appears.

13 Change the URL displayed to reflect the details of your machine.

For example:

```
http://localhost/CustAppSvcs_enu/
start.swe?SWEExtSource=CustomUI&SWEExtCmd=Execute&WSSOAP=1
```

14 Click Next.

The wizard creates the new integration object, writes any errors to a log file, and displays the name and location of the log file.

15 On the Integration Object Builder page, click Finish.

The Web UI DDK Wizard uses the *TestAccountIO* integration object and *TestAccountBS* UI Data Sync service to publish a Web Service that can be used to expose the All Account List (SCW) view. The new Web Service, *TestAccountBS*, is now displayed in the Available Web Services list on the Web Services Data Source page.

16 Select the *TestAccountBS* Web Service, and then click Next.

The Web UI DDK Options page appears, prompting you to define the location of the output Web UI DDK files and displaying details of the target platform, tag library, and proxy class generator.

17 Enter a location for the Web UI DDK output files, such as c:\MyDDKFiles, and then click Finish.

The Web UI DDK Wizard uses the integration object and Web Service to generate the sample JSP pages, Java source code, and WSDL files that represent the All Account List (SCW) view. The output files are saved in the specified location and the wizard closes.

To verify results of Web UI DDK Wizard

1 In Siebel Tools, select the Business Service object type in the Object Explorer.

2 Navigate to the *TestAccountBS* business service and verify the object properties and Business Service Methods.

3 Navigate to the *TestAccountIO* integration object and verify the object properties and child integration components.

4 Compile the *MyAccountProject* project to the server SRF.

5 Start the Siebel Windows Services (Gateway Server and Siebel Server) by opening the Services control panel and starting Siebel Services.

6 After the server starts, open the Siebel Sales application in a browser that points to the Server started in the previous step, and then log in using the following user credentials:

- Username: SADMIN
- Password: SADMIN

7 Navigate to the Administration - Web Services screen, then the Inbound Web Services view.

8 In the Inbound Web Services applet, query for the *TestAccountBS* Web Service to verify that it was published.

9 Open Windows Explorer, navigate to the DDK output directory you specified (c:\MyDDKFiles), and verify the output files appeared.

The DDK output files are ready to hand off to the Java developer to deploy in a custom-built Web application. For deployment examples, see the following:

- "Example of Deploying Java Artifacts (IBM WebSphere)" on page 35
- "Example of Deploying Java Artifacts (Oracle WebLogic)" on page 37

Example of How the DDK Wizard Creates Dynamic Picklists Without a Pick Applet

This topic gives one example of how the Web UI Dynamic Developer's Kit (DDK) Wizard creates a dynamic picklist for a view in the user interface when a business component field does not have a pick applet defined. It also shows how users can use this picklist in the Siebel application to choose from a given set of values. For more information about picklists without pick applets, see "Dynamic Picklists Without Pick Applets" on page 20 in the "About the Sample JavaServer Pages" topic.

NOTE: The DDK generates dynamic picklists without pick applets in a manner similar to Java artifact generation, but the picklists for the JavaServer pages are automatically created and is transparent to developers and users. It is recommended you first familiarize yourself with how Java artifacts are generated before reviewing this example. For more information about generating Java artifacts, see "Process of Generating Java Artifacts" on page 23.

Example of How the Web UI DDK Wizard Creates a Currency Picklist

This example describes how the Web UI DDK Wizard generates Java artifacts when a control on an applet has a dynamic picklist defined for its underlying business component field but does not have a picklist applet defined for that field. Specifically, this example shows how the Web UI DDK Wizard automatically creates a Currency picklist by generating sample JavaServer pages for the Opportunity Detail - Contacts View view and how users can change the currency value using that picklist.

To create a Currency picklist for the Opportunity Detail - Contacts View view, the Web UI DDK must first confirm that no pick applet is configured for one of the controls (with dynamic picklist defined in the underlying business component's field) on the form applet for this view. In this example, the Pick Applet property for the Opportunity Currency control is blank (on the Opportunity Form Applet - Child applet for the Opportunity Detail - Contacts View view).

The DDK Wizard then:

1. Assigns an applet to the view.

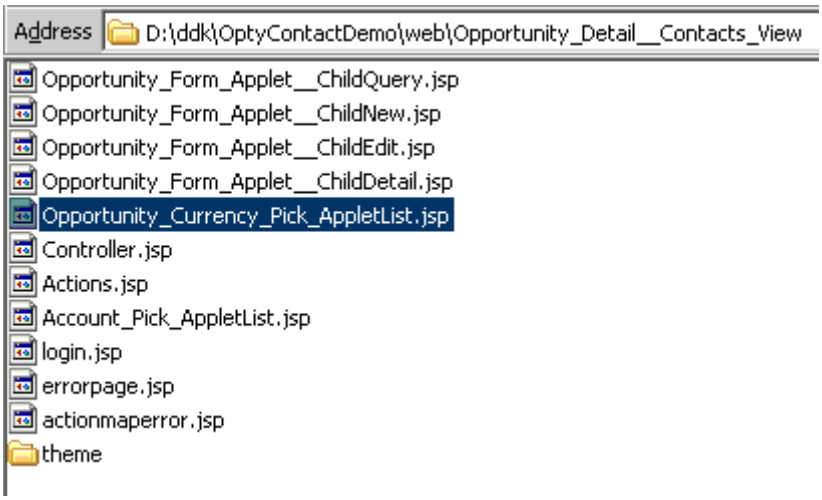
   In this example, the Opportunity Form Applet - Child applet is assigned to the Opportunity Detail - Contacts View.

2. Selects a Web Services data source.

   In this example, WC_Opportunity_BS is selected as the data source.

3 Generates several files containing the Java EE artifacts to run the Mobile Web Client.

In this example, an Opportunity_Currency_Pick_AppletList.jsp file, among others, is generated and stored in the D:\ddk\OptyContactDemo\web directory as shown below.



The Opportunity_Currency_Pick_AppletList.jsp file is the imitation pick applet that is automatically generated for the Opportunity Currency control and is used to show all possible values for that control. To view the value choices, see Figure 4 on page 31.

Keep in mind the steps above are transparent to the user; a clickable Currency field is automatically generated when the Web page renders. In the user interface, the user can change the currency value by clicking the icon next to the Currency field (see "Changing the Currency Value" on page 30).

Changing the Currency Value

The generation of the JavaServer pages for dynamic picklists is automatically created for the user in the user interface. The user sees a Currency field with a clickable icon next to it as shown in Figure 3.

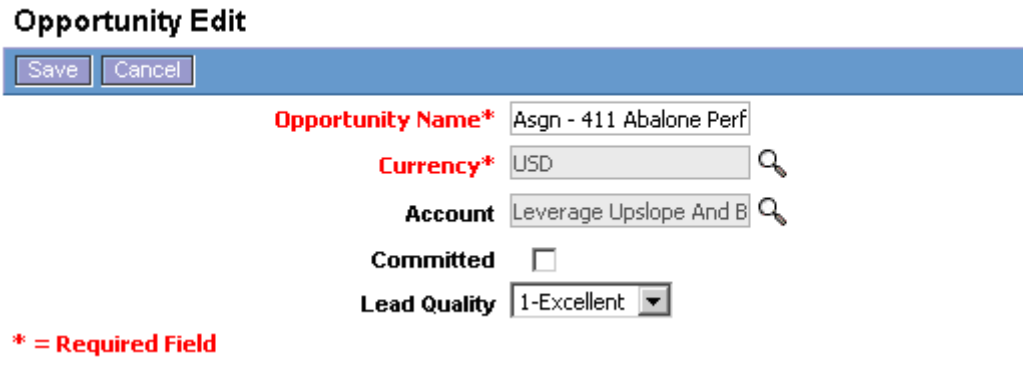


Figure 3. Currency Field Showing Clickable Icon

In this example, the user clicks the icon next to the Currency field to show the choices available from the Currency picklist. The choices originate from the Opportunity_Currency_Pick_AppletList.jsp file generated in Step 3 above and are shown in Figure 4.

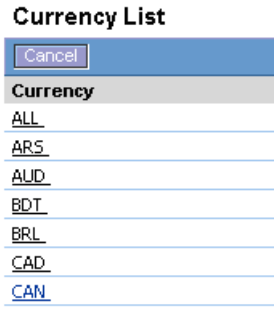


Figure 4. Currency Picklist

In this example, the user selects “CAN” for Canada.

After the selection is made, the new Currency value appears as shown in Figure 5.

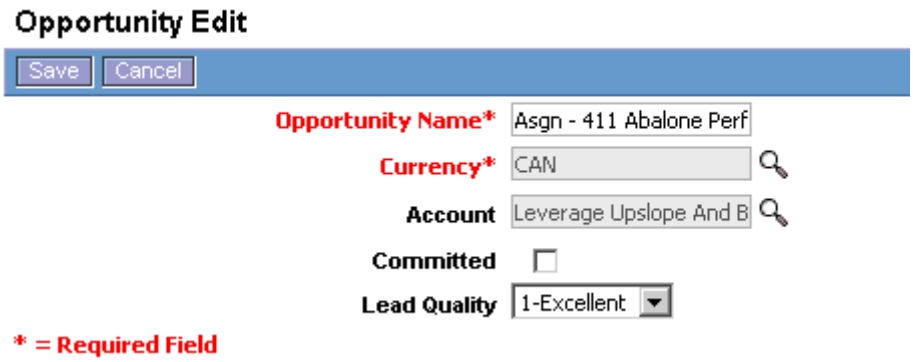


Figure 5. Currency Field Showing New Currency Selection

4 Deploying Java Artifacts

This chapter includes the following topics:

Process for Deploying Java Artifacts

To deploy the Java artifacts generated by the Web UI DDK Wizard, complete the following tasks:

1. "Deploying Compiled SRF" on page 33
2. "Migrating Web Services Definitions" on page 33
3. "Enabling the Custom Application Object Manager Server Component" on page 34
4. "Using the Readme File to Deploy Sample JavaServer Pages" on page 34

Deploying Compiled SRF

Running the Web UI DDK Wizard makes changes to repository objects that you must compile to the SRF file before the changes will take effect. Because you typically run Siebel Tools in a development environment, be sure to migrate the SRF to any other environment in which you want to deploy the sample pages.

This task is a step in the "Process for Deploying Java Artifacts" on page 33.

Migrating Web Services Definitions

Typically, you will be working in a development environment when using Siebel Tools to generate sample JSP pages and other Java artifacts. Web Service data that is created using the Web UI DDK Wizard is stored as administration data in the database's run-time tables. You need to export Web Service definitions from the development environment and import them into the environment in which you want to deploy the sample JavaServer pages.

This task is a step in the "Process for Deploying Java Artifacts" on page 33.

To migrate Web Service definitions

1. Make sure you have compiled the SRF and deployed it on the Siebel Server in your target environment.
2. In the development environment, open the Siebel Web Client.
3. Navigate to the Administration - Web Services screen, then the Inbound Web Services view.
4. Select the Web Service, and then click Export.
5. In the Save As dialog, select the location where you want to store the file.

   Accept the default name and file type (XML).
6. In the target environment (test or production), open the Siebel Web Client.
7. On the application-level menu, choose Navigate, Site Map, Administration - Web Services, and then Inbound Web Services.
8. In the Inbound Web Services applet, click Import.
9. Select the XML file you exported, and then click import.
10. Change the status of the inbound Web Service to active.

Enabling the Custom Application Object Manager Server Component

The Custom Application Object Manager is a server component in the Enterprise Application Integration (EAI) component group. Before you deploy the sample JavaServer pages, make sure that the EAI component group is enabled and the Custom Application Object Manager component is enabled and online.

Server component groups can be enabled when you initially configure of the Siebel Server or any time after, using Server Manager. For more information about configuring and administering server component groups and server components, see *Siebel System Administration Guide*.

This task is a step in the "Process for Deploying Java Artifacts" on page 33.

Using the Readme File to Deploy Sample JavaServer Pages

The steps for deploying the sample JavaServer pages generated by the Web UI DDK Wizard are detailed in the readme.txt file in the DDK output folder. It includes detailed tasks for deploying the sample JavaServer pages in IBM WebSphere Studio Application Developer and Oracle Workshop for WebLogic. Tasks include:

- Copying files from the DDK output folder to appropriate folders in the target Java EE development environment.
- Generating proxy classes from WSDL files.

- Modifying Config.java file to specify a URL that overrides the URL defined in the WSDL files.

This task is a step in the "Process for Deploying Java Artifacts" on page 33.

Example of Deploying Java Artifacts (IBM WebSphere)

This topic gives one example of deploying the Web UI DDK. You may use this feature differently, depending on your business model.

This example assumes that Siebel Web Client and the Siebel Server are installed and running (with the Custom Application Object Manager server component enabled and online) and that the Java EE development environment is IBM WebSphere Studio Application Developer. It also assumes that the Web UI DDK was generated for a view, as described in "Example of Generating Java Artifacts" on page 26.

To deploy Java artifacts in WebSphere

1. Start WebSphere Studio Application Developer (WSAD).
2. After WSAD has initialized, create a new project by selecting File, New, and then the Project menu item.

   The New Project dialog box appears.
3. Select Web in the left text box.

   Two project types display in the right text box.
4. Select Dynamic Web Project and then click Next.
5. Enter a project name, such as MyWSADProject, and then click Finish.

   A message box appears, asking whether you wish to switch to the Web Perspective.
6. Click Yes.

   The new project appears in the left window.

   NOTE: In the left button bar, the Globe button is selected. This means that you are in the Web perspective. Keep this button selected throughout this procedure. If you accidentally change your screen perspective, click the Globe button to restore it.

7 From the DDK output directory you created in "Example of Generating Java Artifacts" on page 26 (*c:\MyDDKFiles*), copy the files from the Web UI DDK output folder and paste them into the WSAD directories as shown in Table 5.

You can paste files into the WSAD project tree by right-clicking the folder you want and then selecting paste from the pop-up menu.

Table 5. Folders to Paste from Web UI DDK to WSAD

Copy These Folders from the Web UI DDK Output Folder	Paste Files into this Folder of the WSAD Project Tree
lib	WebContent/WEB-INF/
wsdl	root directory
allaccountsview\src\com	Java Resources
allaccountsview\web\All_Accounts_List_View	WebContent

NOTE: In the four com.siebel Java packages under the Java Source folder, red 'x' boxes appear against some of the items. This means that errors have been detected. At this stage, the errors are not an issue.

8 Generate the Java client proxies from the WSDL file by, right-clicking on the build.xml file in the wsdl folder of the Web UI DDK, and then selecting Run Ant.

The build dialog displays.

9 Click the Run button.

You will see various messages appear as the Ant process reads the WSDL file and generates the necessary Java client proxies.

10 Select the root of the project tree in WSAD, right-click, and select refresh from the popup menu.

This will refresh the tree from disk with files created in the previous step.

Under the Java Resources folder you can several new packages appear (outputs of the Ant process) appear and the red 'x' boxes have disappeared.

11 Compile the whole project by right-clicking on the root folder of the project and then selecting the Rebuild Project menu item.

Run the application by right-clicking the Login.jsp file in the WebContent/All_Accounts_List_View folder in WSAD.

WSAD installs the new project in the WebSphere application server. After the WebSphere application server has initialized the screen switches to a Web browser and displays the login page.

NOTE: At this stage, the browser is communicating with WebSphere only. No communication between WebSphere and the Siebel application has taken place.

12 Make sure the Siebel Server is running and the Custom Application Object Manager server component enabled and online.

13 Enter a valid username and password, such as SADMIN / SADMIN, and then click Sign In.

The All Account List View appears. You can navigate through the view, create, update, and delete records. Login to the Siebel Web Client to verify that changes to the data are real time.

Example of Deploying Java Artifacts (Oracle WebLogic)

This topic gives one example of deploying Web UI DDK using Oracle WebLogic. You may use this feature differently, depending on your business model.

This example assumes that Siebel Web Client and the Siebel Server are installed and running (with the Custom Application Object Manager server component enabled and online) and that the Java EE development environment is Oracle Workshop for WebLogic. It also assumes that the Web UI DDK was generated for a view, as described in "Example of Generating Java Artifacts" on page 26.

To deploy Java artifacts in Oracle Workshop for WebLogic

1 In the WSDL folder of Web UI DDK output folder, run wsdl2java.bat batch file to generate Java client proxies.

A new com folder containing java proxy classes is created inside the WSDL folder.

The wsdl2Java.bat file is a batch file that invokes the Axis proxy generator.

NOTE: The DDK output directory is specified during the process of running the DDK Wizard. See "Example of Generating Java Artifacts" on page 26.

2 Start WebLogic Application Workshop.

3 After WebLogic has initialized, create a new Application by selecting File, New, and then the Application menu item.

The New Application dialog box appears.

4 Select All Application in the left text box.

Six application types display in the right text box.

5 Select Default Application.

6 Enter a application name, such as *MyApplication*, and then click Create.

A new Web application is created with 2 projects: a schema project and a MyApplicationWeb project.

7 Import files from the Web UI DDK output folder into to the appropriate folder in WebLogic as shown in Table 6.

You can import files into WebLogic by right-clicking a folder, selecting Import from the pop-up menu, and then navigating to the files you want to import

Table 6. WebLogic Folders and Web UI DDK Files

Select this Folder in the WebSphere Application	Import Files from This Folder in the Web UI DDK Output Folder
MyApplicationWeb\WEB-INF\lib	lib/*
MyApplicationWeb\WEB-INF\src	allaccountsview\src\com
MyApplicationWeb\WEB-INF\src	\wsdl\com
MyApplicationWeb folder	web\ All_Accounts_List_View

NOTE: MyApplication is the name of the application created in Step 6.

8 Select the Build menu and then select the Build Application option.

The build dialog displays saying build successful.

9 Select the Tools Menu and then select Weblogic Server/Start Weblogic Server option.

Various messages appear as the WebLogic server starts running. The RED dot in the bottom status bar turns to green.

10 Make sure the Siebel Server is running and the Custom Application Object Manager server component is enabled and online.

11 Open a new browser and type in the following URL:

`http://localhost:7001/MyApplicationWeb/All_Account_List_View/login.jsp`

The browser window refreshes and displays the login page.

12 Enter a valid User Name/Password, such as SADMIN/SADMIN, and then click Sign In.

The All Account List View is displayed. Take some time to navigate through the various screens, create, update and delete records. You can check in the regular Siebel client to see your changes to the data.

5 Using Native Web Service Technology Stacks

Instead of using the sample JSP pages and Java code generated by Web UI DDK Wizard, which require Apache Axis, you can build your own Java client and JSP pages using native Web service technology stacks from IBM WebSphere or Oracle WebLogic. In this case, you would use native tools to generate proxy objects from the WSDL files created by the Web UI DDK Wizard and write custom code to implement the Siebel session management and authentication SOAP headers. This chapter covers the following topics:

Generating Java Client Proxies

You can use native Web technology stacks from IBM WebSphere or Oracle WebLogic to generate Java client proxies from the WSDL files generated by the Web UI DDK Wizard.

To generate Java client proxies in IBM WebSphere

1. Create a new Dynamic Web Project.
2. Import custom Java classes for handling authentication and session management SOAP headers.

   For example Java classes, see "Examples of Setting and Getting SOAP Headers Using IBM WebSphere" on page 41.
3. Import the WSDL file(s) from the Web UI DDK output folder into the Dynamic Web Project created in step 1.
4. Generate the Java client proxies, by right-clicking on the WSDL file, and then navigating to Web Services, then Generate Client.
5. Select Java Proxy for the Client Proxy Type and then select the Test the Generated Proxy checkbox.

   Proxy files (.java files) are generated into Java Resources of the project and sample JSP files are generated and saved in WebContent, Sample, and then the WSDL named directory.

   NOTE: The proxy files are generated with setEndPoint() and get Endpoint() methods that let the same proxy interact with different implementations.
6. Build the project by right-clicking on the project and then selecting Build Project.

7 To test the Web project, right-click on the `WebContent/Sample/{Service}/TestClient.jsp`, and then select Run On Server.

   For example: `http://localhost:9080/Sample/EAILOVPortProxy/TestClient.jsp`

8 Once the server has started, Method Names become available in the left frame.

To generate Java client proxies in Oracle WebLogic

1 In Oracle Workshop for WebLogic, create a new application.

2 Import custom Java classes for handling Siebel session management and authentication SOAP headers.

   For example Java classes, see "Example of Setting and Getting SOAP Headers Using Oracle WebLogic" on page 49.

3 Create a new directory in the Application Web folder.

4 Import the WSDL file(s) to the directory you created in step 3.

5 Right-click on the WSDL file and then choose Generate Service Control.

   The client proxy (for example EAILOVWS.JCX) is generated.

6 Right-click on the .JCX file and then select Generate Test JWS file (Stateless)

   A Java Web Service file (.JWS) is created.

7 To test the Web project, double click the `<BusinessServiceName>.JCX` file, start the WebLogic Server, and then click the Menu Play button or press CTRL+F5.

   A test browser opens that allows you to use the Test XML or Test Form feature.

Implementing Siebel Session Management and Authentication SOAP Headers

IBM WebSphere and Oracle WebLogic provide native technology stacks for supporting Web service clients. To implement Siebel session management and authentication SOAP headers using these stacks, you need to write custom code for setting and getting the SOAP headers. See the following topics for examples:

- "Examples of Setting and Getting SOAP Headers Using IBM WebSphere" on page 41
- "Example of Setting and Getting SOAP Headers Using Oracle WebLogic" on page 49

Siebel session management and authentication SOAP headers are used to pass user credentials and session information back and forth between the Siebel Server and the custom UI. The two primary scenarios for using Siebel session management and authentication SOAP headers are:

- The custom UI sends a valid username and password to the Siebel Server in the SOAP header request. The Siebel Server returns a SessionToken containing a Session ID in the SOAP header response.

- The custom UI sends a SessionToken (received from a previous login) to the Siebel Server in the SOAP header request. The Siebel Server uses the SessionToken to reconnect to an open session and then returns a new SessionToken in the SOAP header response.

  NOTE: Requests (except for login requests) must include a SessionToken returned in the previous response from the Siebel Server.

Table 7 summarizes the Siebel session management and authentication SOAP headers.

Table 7. Siebel Session Management and Authentication SOAP Headers

Name	Description	Sample Siebel SOAP Headers
SessionType	Type of session (Stateful, Stateless, or Server Managed).	<soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <ns1:SessionType xmlns:ns1="http://siebel.com/webservices">stateless</ns1:SessionType> <ns2:UsernameToken xmlns:ns2="http://siebel.com/webservices">SADMIN</ns2:UsernameToken> <ns3:PasswordText xmlns:ns3="http://siebel.com/webservices">MSSQL</ns3:PasswordText> </soapenv:Header> **NOTE:** Values for SessionType are case insensitive.
UsernameToken	User's Login ID name.	
PasswordText	Password used by the Login ID.	
SessionToken	Siebel generated token (encrypted) that includes Session ID, username, and password.	<siebel-header:SessionToken xmlns:siebelheader="http://siebel.com/webservices">MN9SxI9Any9zGQTOFIuJEJfCXjfI0G-9ZOOH4lJjbSd2P.G7vySzo07sFeJxUA0WhdnK</siebel-header:SessionToken> </SOAP-ENV:Header>

For detailed information about Siebel session management and authentication SOAP headers, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

Examples of Setting and Getting SOAP Headers Using IBM WebSphere

To use Siebel session management and authentication SOAP headers in IBM WebSphere, you need to write custom code to set and get the SOAP headers. The following example demonstrates implementing Siebel SOAP headers using IBM WebSphere. The example has three components:

- "Client Context Handler Example" on page 42
- "MySessionBean Example" on page 44
- "JSP Page Example" on page 45

Client Context Handler Example

This example demonstrates including Java dependencies, declaring UsernameToken, PasswordText, and SessionType as strings, and setting and getting the SOAP headers.

```
package com.siebel.headers.handlers;

import javax.xml.namespace.QName;
import javax.xml.rpc.handler.*;
import javax.xml.rpc.handler.soap.SOAPMessageContext;
import javax.xml.soap.*;
import java.util.Iterator;

public final class ClientContextHandler implements Handler {

       QName qn[] = null;
       public static String uname="username";
       public static String pword="password";
       public static String sessionType = "Stateless";
       //public static String sessionTypeLogout = "None";
       public static String SESSID;
       public static String ACTID;

       private static final String NS_URI
              = "http://siebel.com/webservices";
       private static final String SESSIONTYPE_ELEMENT
              = "SessionType";
       private static final String USERNAME_ELEMENT
              = "UsernameToken";
       private static final String PASSWORD_ELEMENT
              = "PasswordText";
       private static final QName SESSIONTYPE_HEADER
              = new QName(NS_URI, SESSIONTYPE_ELEMENT);
       private static final QName USERNAME_HEADER
              = new QName(NS_URI, USERNAME_ELEMENT);
       private static final QName PASSWORD_HEADER
              = new QName(NS_URI, PASSWORD_ELEMENT);

// Construct SOAP Header Elements Here ***

       public static QName[] HEADERS = new QName[]
{ SESSIONTYPE_HEADER,USERNAME_HEADER,PASSWORD_HEADER };

       public MySessionBean sessionId;
       public void init(HandlerInfo info) {
              qn = info.getHeaders();
              java.util.Map m = info.getHandlerConfig();
              uname = (String) m.get("username");
              pword = (String) m.get("password");
       }

       public boolean handleRequest(MessageContext msgContext) {
                     SOAPMessageContext smc = (SOAPMessageContext)msgContext;
                     SOAPMessage msg = smc.getMessage();
                     SOAPPart sp = msg.getSOAPPart();
```

```
                        try {
                                SOAPEnvelope se = sp.getEnvelope();
                                SOAPHeader header = se.getHeader();

                                // add SessionType

                                Name name1 = se.createName(qn[0].getLocalPart(),
                                        "s1", qn[0].getNamespaceURI());
                                SOAPHeaderElement headerElement1 =
header.addHeaderElement(name1);
                                sessionId =
(MySessionBean)msgContext.getProperty("SessionID");
                                if(sessionId.getValue()!=null)
                                {
                                        headerElement1.addTextNode(sessionType);
                                }
                                //if SessionId is null ie Login does not happen
                                        else
                                        {
        headerElement1.addTextNode("None");
                                        }

                                // add UsernameToken
                                Name name2 = se.createName(qn[1].getLocalPart(),
                                        "s2", qn[1].getNamespaceURI());
                                SOAPHeaderElement headerElement2 =
header.addHeaderElement(name2);
                                headerElement2.addTextNode(uname);
                                // add PasswordText
                                Name name3 = se.createName(qn[2].getLocalPart(),
                                        "s3", qn[2].getNamespaceURI());
                                SOAPHeaderElement headerElement3 =
header.addHeaderElement(name3);
                                headerElement3.addTextNode(pword);
                        } catch (SOAPException e) {
                                e.printStackTrace();
                        }
                        return true;
                }

                public boolean handleResponse(MessageContext msgContext)
                {
                        SOAPMessageContext smc = (SOAPMessageContext)msgContext;
                        SOAPMessage msg = smc.getMessage();
                        SOAPPart sp = msg.getSOAPPart();
                        try {

                                SOAPEnvelope se = sp.getEnvelope();|
                                SOAPHeader header = se.getHeader();

                                Iterator it = header.getChildElements();
                                int level=0;
                                while(it.hasNext())
                                {
                                        SOAPHeaderElement he = (SOAPHeaderElement) it.next();
```

```
                                    Name name1 = he.getElementName();
                                    Iterator it2 = he.getChildElements();
                                    level++;
                                    if (it2.hasNext())
                                    {
                                          if (level<=1)
                                          {
                                                Text t = (Text) it2.next();
                                                sessionId =
(MySessionBean)msgContext.getProperty("SessionID");
                                                sessionId.SetValue(t.getValue());
                                          }
                                          else
                                          {
                                                Text t = (Text) it2.next();
                                          }

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

                        }

                        return true;

                }

                public QName[] getHeaders() {
                        return qn;
                }

                public void destroy() {}

                public boolean handleFault(MessageContext mc) {    SOAPMessageContext
messageContext = (SOAPMessageContext) mc;
                        return true;  }
}
```

MySessionBean Example

This example demonstrates storing the Session ID in the MySessionBean object.

```
package com.siebel.headers.handlers;

public class MySessionBean {

String value = "default";

   public String getValue()

   {

      return value;

   }
```

```
   public void SetValue(String strSessionID)
{
       value = strSessionID;

   }
}
```

JSP Page Example

The following JSP example demonstrates how the HandlerInfo object passes username, password, and session ID into the handler.

```
<HTML>
<HEAD>
<%@ page language="java"
   contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"
   session="true"
%>

<%@ page import="com.siebel.www.*" %>
<%@ page import="com.siebel.headers.handlers.*" %>
<%@ page import="javax.xml.rpc.handler.*" %>
<%@ page import="javax.xml.namespace.QName" %>
<%@ page import="java.util.*" %>
<%@ page import="java.net.URL" %>

<%

   String uname = request.getParameter("username");
   String pword = request.getParameter("password");
   String sessid;

%>

<HEAD>

<TITLE>Siebel Custom Headers</TITLE>
</HEAD>
<BODY>

<FORM METHOD="Get" >
   <TABLE BORDER="0" CELLPADDING="4" CELLSPACING="0" BGCOLOR="#eeeeee">
      <TR BGCOLOR="#ccccff">
         <TD COLSPAN="2" WIDTH="100%">
            <B><FONT FACE="Arial">Sign In</FONT></B>
         </TD>
      </TR>
      <TR>
         <TD ALIGN="right">
            <FONT FACE="Arial" SIZE="-1 ">User ID:</FONT>
         </TD>
         <TD>
            <INPUT NAME="username" SIZE="12" MAXLENGTH="32">
         </TD>
```

```
      </TR>
      <TR>
         <TD ALIGN="right">
            <FONT FACE="Arial" SIZE="-1">Password:</FONT>
         </TD>
         <TD>
            <INPUT NAME="password" SIZE="12" TYPE="password" MAXLENGTH="32">
         </TD>
      </TR>
      <TR>
         <TD>
            <INPUT NAME="action" SIZE="12" TYPE="hidden" value="login">
         </TD>
      </TR>
      <TR>
         <TD>
             
         </TD>
         <TD>
            <FONT FACE="Arial" SIZE="-1"><INPUT TYPE="submit" VALUE="Sign in"
NAME="Invoke"></FONT>
         </TD>
      </TR>
      <TR>
         <TD>
            <INPUT NAME="action" SIZE="12" TYPE="hidden" value="logout">
         </TD>
      </TR>
      <TR>
         <TD>
             
         </TD>
         <TD>
            <FONT FACE="Arial" SIZE="-1"><INPUT TYPE="submit" VALUE="Logout"
NAME="InvokeLogout"></FONT>
         </TD>
      </TR>
   </TABLE>
</FORM>

<%
   if (request.getParameter("Invoke")!= null)
   {
      uname = request.getParameter("username");
      pword = request.getParameter("password");
      sessid = InvokeWS(uname, pword);

      out.println("Your session ID is: " + sessid);
   }

%>

<%
```

```
   if (request.getParameter("InvokeLogout")!= null)
   {

       uname = request.getParameter("username");
       pword = request.getParameter("password");
       sessid = InvokeLogout(uname, pword);

       out.println("Your session ID is: " + sessid);
   }

%>

<%!

MySessionBean ctxValue;
       Port service;
       SessionAccessWSLocator locator;
       List handlerList ;
       HandlerRegistry hInfo;
       Map hConfig ;
public String InvokeWS(String username, String password)
{
   try{

       locator = new SessionAccessWSLocator();

       Iterator i = locator.getPorts();
       QName temp =(QName) i.next();

       //QName portQName= new QName("http://tempuri.org/", "WSHeaderSoap");
       //List handlerList = new ArrayList();
       handlerList = new ArrayList();
       HandlerRegistry hInfo = locator.getHandlerRegistry();
       hInfo = locator.getHandlerRegistry();

       hConfig = new HashMap();
       hConfig.put ("username", username);
       hConfig.put ("password", password);

       handlerList.add(new HandlerInfo(ClientContextHandler.class, hConfig,
ClientContextHandler.HEADERS));
       hInfo.setHandlerChain(temp, handlerList);

       //WSHeaderSoapProxy wsheaderproxy = new WSHeaderSoapProxy();
       //WSHeaderSoap getConversationtemp = locator.getWSHeaderSoap();
       service = locator.getPort();

       ctxValue = new MySessionBean();
       ctxValue.SetValue("no session");
       //((WSHeaderSoapStub)getConversationtemp)._setProperty("SessionID",ctxValue);
       ((PortStub)service)._setProperty("SessionID", ctxValue);

       //getConversationtemp.doInvokeHeaders();
       service.sessionAccessPing(null);
```

```
      ctxValue = (MySessionBean)((PortStub)service)._getProperty("SessionID");
      //System.out.println("Session Bean is " + ctxValue.getValue());
      return (ctxValue.getValue());

   }
   catch(Exception e)
   {
      e.printStackTrace ();
      System.out.println (e.getMessage ());
      return (e.toString());
   }
}

%>

<%!

public String InvokeLogout(String username, String password)

{

   try{

      //SessionAccessWSLocator locator = new SessionAccessWSLocator();
      Iterator i = locator.getPorts();
      QName temp =(QName) i.next();
      //QName portQName= new QName("http://tempuri.org/", "WSHeaderSoap");
      //List handlerList = new ArrayList();
      handlerList = new ArrayList();
      //HandlerRegistry hInfo = locator.getHandlerRegistry();
      hInfo = locator.getHandlerRegistry();

      //Map hConfig = new HashMap();
      //hConfig.put ("username", username);
      //hConfig.put ("password", password);

      handlerList.add(new HandlerInfo(ClientContextHandler.class, hConfig,
ClientContextHandler.HEADERS));
      hInfo.setHandlerChain(temp, handlerList);

      //WSHeaderSoapProxy wsheaderproxy = new WSHeaderSoapProxy();
      //WSHeaderSoap getConversationtemp = locator.getWSHeaderSoap();|
      service = locator.getPort();

      //MySessionBean ctxValue = new MySessionBean();
      ctxValue.SetValue(null);
      //((WSHeaderSoapStub)getConversationtemp)._setProperty("SessionID",ctxValue);
      //((PortStub)service)._setProperty("SessionID", ctxValue);
      ((PortStub)service)._setProperty("SessionID", ctxValue);

      //getConversationtemp.doInvokeHeaders();
      service.sessionAccessPing(null);

      ctxValue = (MySessionBean)((PortStub)service)._getProperty("SessionID");
      //System.out.println("Session Bean is " + ctxValue.getValue());
      return (ctxValue.getValue());
```

```
    }
    catch(Exception e)
    {
        e.printStackTrace ();
        System.out.println (e.getMessage ());
        return (e.toString());
    }
}

%>

</BODY>
</HTML>
```

Example of Setting and Getting SOAP Headers Using Oracle WebLogic

The following example demonstrates how to set and get Siebel session management and authentication SOAP headers using Oracle Web Logic.

```
    public class SessionAccessControlTest implements com.bea.jws.WebService
    {
        static final long serialVersionUID = 1L;

        /** @common:control */
        public WSDLFolder.SessionAccessControl sessionAccessControl;
    /**
             * @common:operation
             */
             public MySessionBean Login(String Username,String Password)
             {
                 MySessionBean sessiontok=new MySessionBean();

                 HeaderDocument hd = null;
                 try
                 {
                         hd = HeaderDocument.Factory.parse(
                         "<SOAP-ENV:Header xmlns:SOAP-ENV=\"http://
    schemas.xmlsoap.org/soap/envelope/\">" +
                         "<ns1:SessionType xmlns:ns1=\"http://siebel.com/
    webservices\">Stateless</ns1:SessionType>" +
                         "<ns2:UsernameToken xmlns:ns2=\"http://siebel.com/
    webservices\">"+Username+"</ns2:UsernameToken>" +
                         "<ns3:PasswordText xmlns:ns3=\"http://siebel.com/
    webservices\">"+Password+"</ns3:PasswordText>" +
                         "</SOAP-ENV:Header>");
                 Element h1 = (Element) hd.newDomNode().getFirstChild();
                 sessionAccessControl.setOutputHeaders(new Element[] { h1});

                 }
                 catch (XmlException xe)
                 {
```

```
                System.out.println(xe.toString());
            }
            sessionAccessControl.SessionAccessPing("");
            Element e[]=sessionAccessControl.getInputHeaders();
            sessiontok.SetValue(e[0].getFirstChild().getFirstChild().getNodeValue
());
            System.out.println("sESSION id "+ sessiontok.getValue());

            return sessiontok;
    }

}
```

6 About Preconfigured Web Services

A number of preconfigured Web services are delivered as seed data with Siebel Business Applications. They allow external Web applications access to Siebel data and functionality. This chapter describes these services. Topics include:

About UI Data Sync Services

UI Data Sync (UDS) Web services provide external Web applications access to Siebel data and functionality. UDS services expose version-independent interfaces with strongly typed arguments and provide access to UI operations such as insert, delete, update, and query.

UDS Web services are based on UDS business services. UDS business services are defined using the EAI UI Data Service class (CSSEAIUIDataService) and invoke the UI Data Adapter business service (CSSEAIUIDataAdapter). UDS business services convert strongly typed arguments into arguments that the UI Data Adapter business service can process and the UI Data Adapter business exposes methods for manipulating Siebel data.

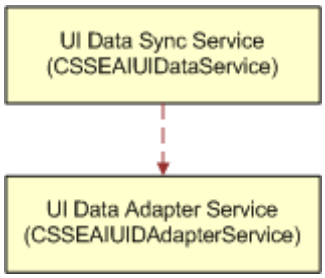


Figure 6. UDS Services and UDA Service

For more information about the UI Data Adapter business service, including descriptions of methods and method arguments, see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

Preconfigured UDS services exist for common objects such as Account, Contacts, Opportunity, and Orders. Preconfigured Web services and their corresponding UDS business services are listed in Table 8.

NOTE: Both UDS business services and Web services are inactive by default. Use Siebel Tools to activate the UDS business services and then recompile the SRF. Use the Siebel Web Client to activate the UDS Web services. For more information see, "Activating Preconfigured Integration Objects and Business Services" on page 24 and "Activating Preconfigured Web Services" on page 24, respectively.

Table 8. Preconfigured UDS Services

Preconfigured UDS Business Services	Preconfigured UDS Web Services
WC_Account_BS	WC_Account_BS
WC_Asset_DB_BS	WC_Asset_DB_BS
WC_Contacts_BS	WC_Contacts_BS
WC_MVG_Channel_Partner_BS	WC_MVG_Channel_Partner_BS
WC_MVG_Household_BS	WC_MVG_Household_BS
WC_MVG_Industry_BS	WC_MVG_Industry_BS
WC_MVG_Internal_Division_BS	WC_MVG_Internal_Division_BS
WC_MVG_Organization_BS	WC_MVG_Organization_BS
WC_MVG_Position_BS	WC_MVG_Position_BS
WC_MVG_Source_BS	WC_MVG_Source_BS
WC_MVG_Territory_BS	WC_MVG_Territory_BS
WC_Opportunity_BS	WC_Opportunity_BS
WC_Orders_BS	WC_Orders_BS
WC_PickList_Abs_Admin_Service_Region_BS	WC_PickList_Abs_Admin_Service_Region_BS
WC_PickList_Action_BS	WC_PickList_Action_BS
WC_PickList_Agreement_Entitlement_BS	WC_PickList_Agreement_Entitlement_BS
WC_PickList_Auction_Services_BS	WC_PickList_Auction_Services_BS
WC_PickList_Business_Address_BS	WC_PickList_Business_Address_BS
WC_PickList_Campaign_BS	WC_PickList_Campaign_BS
WC_PickList_Currency_BS	WC_PickList_Currency_BS
WC_PickList_Employee_BS	WC_PickList_Employee_BS
WC_PickList_FS_Shipping_BS	WC_PickList_FS_Shipping_BS
WC_PickList_Internal_Product_BS	WC_PickList_Internal_Product_BS
WC_PickList_Order_Entry_BS	WC_PickList_Order_Entry_BS

Table 8. Preconfigured UDS Services

Preconfigured UDS Business Services	Preconfigured UDS Web Services
WC_PickList_Payment_Profile_BS	WC_PickList_Payment_Profile_BS
WC_PickList_Payment_Terms_BS	WC_PickList_Payment_Terms_BS
WC_PickList_Price_List_BS	WC_PickList_Price_List_BS
WC_PickList_Sales_Assessment_BS	WC_PickList_Sales_Assessment_BS
WC_PickList_Sales_Stage_BS	WC_PickList_Sales_Stage_BS
WC_PickList_Service_Agreement_BS	WC_PickList_Service_Agreement_BS
WC_PickList_Siebel_Instance_BS	WC_PickList_Siebel_Instance_BS
WC_PickList_VORD_Network_Asset_BS	WC_PickList_VORD_Network_Asset_BS
WC_Quote_BS	WC_Quote_BS
WC_Service_Request_BS	WC_Service_Request_BS

About the LOV Service

The LOV service provides external Web applications the ability to query for lists of values (LOV) from the Siebel database. The published Web service is named EAILOVWS and is based on the EAI LOV Service business service.

The LOV service exposes one method, GetListOfValues. Table 9 describes the method and the method arguments.

Table 9. LOV Service Methods and Method Arguments

Method	Description	Arguments		
		Type	Name	Description
GetListOfValues	Retrieves lists of values to display in picklist fields.	Input	ListsTopElementQuery	Takes the List Query integration object as input
		Output	ListsTopElementResult	Returns the List Result integration object

Table 10 summarizes the hierarchy of the List Query and List Result integration objects.

Table 10. List Query and List Result Integration Object Hierarchy

Integration Object		Integration Object Component		Integration Object Component Fields	
Name	XML Tag	Name	XML Tag	Name	XML Tag
List Query	ListQuery	List	ListQuery	Active	Active
				Filter Value	FilterValue
				Language Code	LanguageCode
				Type	Type
List Result	ListsResult	List	ListResult	Type	Type
		List Value	ListValueResult	Active	Active
				Description	Description
				Display Value	DisplayValue
				Filter Value[1]	FilterValue
				Language Code	LanguageCode
				Order	Order
				Value	Value

1. You cannot use the Filter Value field for searching.

Example of an LOV Service Request

The following example request demonstrates invoking the LOV business service to query for two LOV types, TRAINING_LOC_REGION and MR_MS.

```
<soapenv:Body>
   <EAILOVService_EAILOVGetListOfValues_Input xmlns="http://www.siebel.com/xml/
LOVService">
      <ns3:ListsQuery xmlns:ns3="http://www.siebel.com/xml/ListQuery">
         <ns3:ListQuery>
            <ns3:Type>TRAINING_LOC_REGION</ns3:Type>
         </ns3:ListQuery>
         <ns3:ListQuery>
            <ns3:Type>MR_MS</ns3:Type>
         </ns3:ListQuery>
      </ns3:ListsQuery>
   </EAILOVService_EAILOVGetListOfValues_Input>
</soapenv:Body>
```

Example of an LOV Service Response

The following example shows the response from the LOV Service to the query for two LOV types, TRAINING_LOC_REGION and MR_MS.

```
    <SOAP-ENV:Body>
      <ns:EAILOVService_EAILOVGetListOfValues_Output xmlns:ns="http://
www.siebel.com/xml/LOVService">
        <ListsResult xmlns="http://www.siebel.com/xml/ListResult">
          <ListResult>
            <Type>TRAINING_LOC_REGION</Type>
            <ListValuesResult>
              <ListValueResult>
                <Active>Y</Active>
                <Description></Description>
                <DisplayValue>Asia</DisplayValue>
                <FilterValue></FilterValue>
                <LanguageCode>ENU</LanguageCode>
                <Order>3</Order>
                <Value>Asia</Value>
              </ListValueResult>
              <ListValueResult>
                <Active>Y</Active>
                <Description></Description>
                <DisplayValue>Europe</DisplayValue>
                <FilterValue></FilterValue>
                <LanguageCode>ENU</LanguageCode>
                <Order>2</Order>
                <Value>Europe</Value>
              </ListValueResult>
              <ListValueResult>
                <Active>Y</Active>
                <Description></Description>
                <DisplayValue>Latin America</DisplayValue>
                <FilterValue></FilterValue>
                <LanguageCode>ENU</LanguageCode>
                <Order>4</Order>
                <Value>Latin America</Value>
              </ListValueResult>
            </ListValuesResult>
          <ListResult>
            <Type>MR_MS</Type>
            <ListValuesResult>
              <ListValueResult>
                <Active>Y</Active>
                <Description></Description>
                <DisplayValue>Dr.</DisplayValue>
                <FilterValue></FilterValue>
                <LanguageCode>ENU</LanguageCode>
                <Order>4</Order>
                <Value>Dr.</Value>
              </ListValueResult>
              <ListValueResult>
                <Active>Y</Active>
                <Description></Description>
```

```
                <DisplayValue>Miss</DisplayValue>
                <FilterValue></FilterValue>
                <LanguageCode>ENU</LanguageCode>
                <Order>1</Order>
                <Value>Miss.</Value>
              </ListValueResult>
            </ListValuesResult>
          </ListResult>
        </ListsResult>
      </ns:EAILOVService_EAILOVGetListOfValues_Output>
    </SOAP-ENV:Body>
```

About the Session Access Service

The Session Access Web service provides external Web applications access to operations for starting sessions and accessing profile attributes associated with a session. The published Web service is named SessionAccessWS and is based on the Session Access Service business service.

Table 11 lists the methods exposed by the Session Access Web service and lists the arguments used with each method.

Table 11. SessionAccessService Methods and Arguments

Method Name	Description	Arguments		
		Type	Name	Description
Ping	Returns a fixed value. Used as a null operation	Input	MsgIn	Optional. Ignored if set.
		Output	MsgOut	A fixed string value of “1”.
Echo	Returns the input value as the output value.	Input	MsgIn	Required. Any arbitrary string input.
		Output	MsgOut	Copy of the MsgIn argument.
GetProfileAttr	Retrieves the value of a specified profile attribute from the current object manager session.	Input	Name	Required. Name of the profile attribute.
		Output	Value	Value of the profile attribute.
SetProfileAttr	Sets the value of a specified profile attribute	Input	Name	Required. Name of the profile attribute.
		Input	Value	Value of the profile attribute.

Example of a Session Access Service Request

The following example demonstrates invoking the SessionAccess service and using the Ping method.

NOTE: For more information about Oracle's Siebel session management and authentication SOAP headers shown in the following example (SessionType, UsernameToken, and Password Text), see *Integration Platform Technologies: Siebel Enterprise Application Integration*.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
   <soapenv:Header>
      <ns1:SessionType soapenv:mustUnderstand="0" xsi:type="xsd:string"
xmlns:ns1="http://siebel.com/webservices">Stateless</ns1:SessionType>
      <ns2:UsernameToken soapenv:mustUnderstand="0" xsi:type="xsd:string"
xmlns:ns2="http://siebel.com/webservices">SADMIN</ns2:UsernameToken>
      <ns3:PasswordText soapenv:mustUnderstand="0" xsi:type="xsd:string"
xmlns:ns3="http://siebel.com/webservices">MSSQL</ns3:PasswordText>
   </soapenv:Header>
   <soapenv:Body>
      <SessionAccessService_SessionAccessPing_Input xsi:nil="true" xmlns="http://
www.siebel.com/xml/SessionAccess"/>
   </soapenv:Body>
</soapenv:Envelope>
```

Example of a Session Access Service Response

The following example shows an example response from the Session Access Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Header>
   <siebel-header:SessionToken xmlns:siebel-header="http://siebel.com/
webservices">HwopiOF3X.iNGKb9YCsDLJT1JxtOv.CCSOjsHIWHbX.nvGleXekmiIDHbSXQHG0zXlDaP
cc4UeX66l.k-eVqEpjWhjq4qglvWd1dWXxJderIbn2rYAQTZynx1XD9JmsMZHDgRbp2iMKjad-
RwtrCPVY7IlH6bGmmrVWCX3MyPIY_</siebel-header:SessionToken>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns:SessionAccessService_SessionAccessPing_Output xmlns:ns="http://
www.siebel.com/xml/SessionAccess">
      <ns:MsgOut>1</ns:MsgOut>
      </ns:SessionAccessService_SessionAccessPing_Output>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Index