1 Introduction

This document provides a guide to accessing the Siebel Mobile Connector application programming interface (API) and working with its new business services to create mobile applications using Siebel 7.5. Siebel Mobile Connector is a new, separately licensed component of Siebel 7.5 that enables partners and customers to create voice, wireless and other applications with Siebel eBusiness content.

In this section, you will find an overview of Siebel Mobile Connector and a brief explanation of its architecture. Additionally, scenarios are given to provide understanding of Siebel Mobile Connector from the point of view of the users of partner applications. Finally, there is information about this guide and additional documentation.

About Siebel Mobile Connector

Siebel Mobile Connector enables partners and customers to create voice, wireless and other applications with Siebel eBusiness content. Siebel Mobile Connector is a standards-based API delivering well-formed XML from an optimized Siebel application definition. This gives the calling application a definition of user interface and user data in XML format.

Siebel partner and customer application developers can give mobile users real-time or near real-time access to critical Siebel eBusiness information through a variety of mobile devices. By using speech or a wireless application on a mobile phone or personal digital assistant, users can view, edit and create information in their companies' Siebel eBusiness respository. Siebel data and data from other applications can be combined in the same user interface. For example, employees, partners and customers can:

- Update sales opportunities
- Search for account information
- Access calendar and contact details
- Review order and parts status
- Respond to service requests

The Metadata Business Service enables customers and partners to easily filter data offered through the pre-configured application definitions without having to use Siebel Tools to permanently change the application configuration. Siebel Mobile Connector generates style sheets to limit the data returned from an applet. The style sheets are stored as XSL documents on the Siebel Application Server. The pre-built Siebel Mobile Connector application definition has been optimized for mobile applications. These optimized views are aimed at improving performance on mobile devices. Because Siebel Mobile Connector handles the details of abstracting the lower-level data model, it may not be necessary for third-party application developers to re-write their code following an upgrade to Siebel applications. Thus, the upgrade path for applications using Siebel Mobile Connector becomes relatively independent of the Siebel upgrade path.

If you need to customize the application definition beyond applying style sheets, Siebel Tools can be used. Siebel Mobile Connector uses the same development toolset (Siebel Tools), and the same logical data model as all other Siebel eBusiness Applications.

Additionally, the Siebel Mobile Connector provides an Alert Business Service to communicate changes to specific business components. Notifications can be pushed to employees, partners or customers who spend a majority of their time outside the office. This business service creates an XML document that can be pushed to customer or partner-developed mobile applications. For example, the reassignment of a service request from one service technician to another triggers a workflow; in this process the Alert Business Service creates an XML document with the relevant, pre-defined data and sends it to the mobile application, thereby allowing the application to dispatch a wireless message to both parties indicating their reassignment.

Usage Scenarios

This section provides understanding of Siebel Mobile Connector from the point of view of the users of partner applications. These usage scenarios are here for illustrative purposes. Siebel Alliance partners who have built validated solutions can be found listed on Siebel System's web site: http://www.siebel.com.

Sample Sales Voice Scenario

Siebel Mobile Connector can be used to create a real-time voice interface to Siebel data. In the following example, a salesperson is made more effective by using a voice interface to Siebel Sales.

Joelle Zorica is a salesperson. She is currently on her way to visit John Hiatt, a very important customer; however, she's hit a traffic jam. She does not have the customer's phone number with her so she uses a Sales Voice application to access this information. The Sales Voice application uses the Siebel Mobile Connector interface to retrieve data located in a Siebel data repository at the company offices.

1. Joelle calls into the Sales Voice application from a mobile phone.
2. The system greets her with “Hello. Welcome to the Sales Voice application. Please say your user ID number or enter it using the keypad.”
3. Joelle speaks her user ID number.
4. System: “Please enter your PIN.”
5. Joelle speaks her PIN number and the system authenticates her login as a valid user of the Siebel Sales system.
6. System: “You've got new leads. Would you like to go to Opportunities, Contacts, Accounts, or Calendar?”
7. Joelle: “Contacts.”
8. System: “You're in Contacts. What opportunities do you want to look up?”
9. Joelle: “Look up John Hiatt.”
10. System: “John Hiatt is found...”
11. Joelle: “Call John Hiatt.”

12 The Sales Voice system places a call to John Hiatt and logs off Joelle from the system.

Sample Customer Service Voice Scenario

In the following example, a customer is able to be served more efficiently by using a voice interface to Siebel Call Center.

Allan Street's refrigerator needs repair. He has multiple channels for communicating with the refrigerator's manufacturer: telephone, web site, or even a wireless application. In this particular case, Allan interacts with the manufacturer's Call Center and Customer Service Voice Application. The Customer Service Voice Application uses the Siebel Mobile Connector interface to retrieve data from the company's Siebel Call Center application.

1 Allan Street calls the customer support line of the refrigerator's manufacturer to place a service request to have an service technician come repair his refrigerator. This service request is entered into Siebel Call Center, prioritized, and routed to field service centers or dispatchers. Allan receives his service request ticket number so he can check the status of his request.

2 A few hours later, Allan calls into the Customer Service Voice application to check on the status of his service request.

3 The system greets him with "Hello. Welcome to the Customer Service Voice application. Please say your ticket number or enter it using the keypad."

4 Allan speaks his ticket number and the system authenticates his login as an anonymous user. The Customer Service Voice application requests the information for this ticket from the Siebel Call Center application.

5 The system plays back the ticket information: "Your ticket number is 654321. Your request is regarding repair of refrigerator model RF1. Your order is currently assigned for repair tomorrow at 1 P.M..."

6 Allan decides that the description doesn't have enough detail, so he updates the description with the various sounds his refrigerator is making (the description is attached as an audio file). The problem is worse than he thought, so Allan escalates the service request since the repair technician is not scheduled to arrive until tomorrow afternoon.

7 The Customer Service Voice application records Allan's changes and updates the ticket information in the Siebel database. Because the ticket's priority was escalated, a service manager is alerted via e-mail to the change. Allan's changes have triggered an alert condition that is monitored by the Alert Business Service. Aware of the new information that Allan entered into the Customer Service Voice application, the service manager assigns a service technician to pay Allan Street an immediate visit.

8 Allan completes his phone call and the Customer Service Voice application logs him off automatically.

Sample Wireless Sales Scenario

In the following example, a salesperson is made more effective by using a wireless interface to Siebel Sales.

Maria Smith is a salesperson working outside the office. While she is on a sales call, the regional manager assigns Maria an opportunity for a very important prospect. The Wireless Sales application uses the Siebel Mobile Connector interface to retrieve data from the company's Siebel Sales application.

1 Maria receives an SMS message in her personal digital assistant. The message tells her that a new sales opportunity has been assigned to her and is awaiting her action to accept or reject the opportunity. She is able to accept the opportunity through two-way Short Message Service (SMS). However, she wants to get more details.

2 To get more details about the opportunity, Maria enters her login and password into an HTML form on a Web page displayed by a wireless browser running a Wireless Sales application. The system authenticates her login as a valid user of the Siebel Sales system.

3 The system presents her with a Siebel user interface optimized for display in a mobile environment.

4 Maria clicks the Opportunities screen, and a screen is displayed with the data for her opportunities.

5 Maria queries for new Opportunities and finds the new lead assigned to her. She reviews the details of the new prospect, and places a call to the primary contact to begin the sales cycle.

6 When she accepts the opportunity, the Wireless Sales application sends the update for the opportunity to the Siebel database.

Architecture Overview

Siebel Mobile Connector allows application developers to create applications that query for (or pull) information, get information pushed to the application, or create, edit or update information in the Siebel database. The following components are involved in transactions using the Siebel Mobile Connector:

- **A Siebel database.** This is the database that users of the mobile application will access.
- **A Siebel application server.** These components execute all business logic for the Siebel application and provide an XML interface between third-party applications and the Siebel database. The Siebel application server components include the Data Manager, Object Manager, Siebel Web Engine and Siebel Mobile Connector.
- **The Siebel Web Engine.** Siebel Web Engine is a component of the Siebel application server that makes possible the deployment of e-applications in HTML, WML and XML. A Web browser client interacts with the Siebel database through Siebel Web Engine. The Siebel Web Engine contains the XML Web Interface that processes XML requests.
- **Siebel Mobile Connector.** Siebel Mobile Connector contains an optimized application definition, the Alert Business Service, the Metadata Business Service, the GetSMCUpdate method within SWE and the Reference Configuration sample application. Third-party application developers can use Siebel Mobile Connector to access sales, service, or self-service data, create style sheets to filter Siebel data, retrieve updates, generate and send alert (push) workflows. Siebel Mobile Connector uses the XML Web Interface of SWE to retrieve information from the Siebel Mobile Connector application definition (or any other Siebel application definition).
- **Siebel Business Process Administration.** This is a component that enables alerts to be sent to a third-party application server. It is a business application that can be customized by defining and managing the workflows that the alerts are based on.

- **A Reference Configuration Sample.** This sample is provided to show how to access the Metadata Business Service. The sample generates style sheets used by the third-party application and alerts based on defined business processes. It also allows the application developer to configure smcalert.cfg, the configuration file used by the Alert Business Service to determine the transport mechanism. It is hosted by an enterprise on a Microsoft Windows 2000 web server. Third-parties may wish to expose this capability within their own toolset, enabling developers to configure the third party application.

- **A third-party application server.** This middleware application server exposes the infrastructure necessary for building mobile applications between the Siebel application and the end user. The middleware application server is responsible for queries for and retrieval of Siebel data. It interfaces with the Siebel Web Engine XML Web Interface using XML commands and presenting the data as required to the end-user. For example, a voice application server would contain the necessary telephony, speech recognition, and text-to-speech capability to interface with a user via speech application. Additionally, a wireless online/ offline application would contain the necessary client-server queuing software to enable the storage and forwarding of messages from the server to the client or vice versa.

- **A mobile client.** In deployments of applications in wireless environments, Siebel data is accessed by users with client software residing on a mobile device such as a personal digital assistant or mobile phone. The client software is capable of accessing the third-party application and displaying a user interface in HTML, WML or other mark-up language. It is not necessary that the application platform be a mobile device; other platforms can also be used with Siebel Mobile Connector. In the Siebel architecture, no components are hosted on the client.

Figure 1 illustrates the architecture of a system using Siebel Mobile Connector to provide access to the Siebel database from a third-party mobile application.

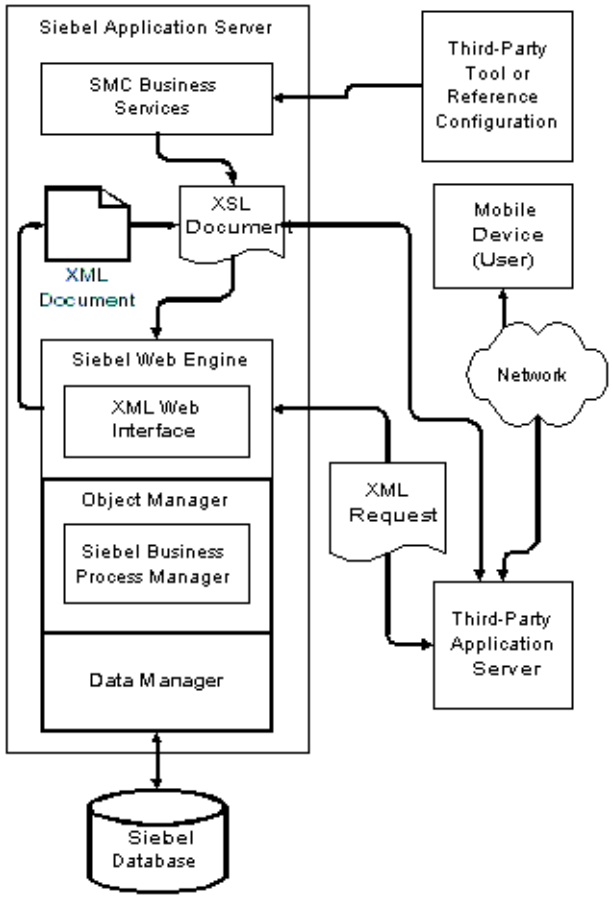


Figure 1. Siebel Mobile Connector Architecture

Each block in this figure represents a separate machine, although some of the components shown separately could be installed on a single machine.

How to Work with Siebel Mobile Connector

Third-party applications created with Siebel Mobile Connector use standard Web protocols or specific Siebel interfaces such as a Java Data Bean or the COM Data Control to send and retrieve data between users and the Siebel database. These steps describe the flow in greater detail and assume that real-time access to Siebel data is available.

1. Using the third-party application's user interface, the user requests information residing in the Siebel database.
2. The third-party application passes the information requested by the user to Siebel Web Engine (SWE) in the form of an XML document. The request is made through HyperText Transfer Protocol (HTTP), Java Data Bean, COM Data Control or any Siebel object interface.
3. Siebel Mobile Connector, which is part of the Siebel application server, invokes SWE to retrieve information from the Siebel database.
4. This information is then passed back through Siebel Object Manager to SWE.
5. When SWE has the requested data, it returns it in the form of an XML document to the third-party application. If less than the total data set is wanted, the request specifies a style sheet that should be applied to the data. The style sheets are located at the Siebel application server
6. The third-party application parses the XML document and presents the Siebel data to the user in its own user interface.

NOTE: If your application provides online/offline capabilities, it must have the capability for storing and forwarding messages. In other words, your application must have a feature that queues messages between server and client, allowing the exchange of messages.

For more detailed information on how the business services of Siebel Mobile Connector work, see the following sections: "How Metadata Business Service Works" on page 90 and "How the Alert Business Service Works" on page 120.

About the Documentation

This section provides information on how the Siebel Mobile Connector Guide and additional documentation available.

How This Guide Is Organized

This book is organized in a way that presents information on Siebel Mobile Connector and its business services as individual chapters. Additional information, including a glossary and troubleshooting assistance, can be found in the appendices.

This book contains the following chapters:

- Chapter , "Installing and Configuring Siebel Mobile Connector."
- Chapter , "Working with Siebel Mobile Connector."
- Chapter , "Working with Metadata Business Service."
- Chapter , "Working with Alert Business Service."
- Chapter , "Troubleshooting."

Additionally, there are appendices that provide a reference to the Siebel Mobile Connector application definition and sample XML and XSL output.

Who Should Use This Guide

This book will be useful primarily to people whose title or job description matches one of the following:

Business Analysts	Persons responsible for analyzing business needs, application integration challenges and planning integration solutions at an enterprise.
Siebel Application Administrators	Persons responsible for planning, setting up, and maintaining Siebel applications.

Siebel Application Developers	Persons responsible for planning, implementing, and configuring Siebel applications, and possibly adding new functionality.
Siebel Integration Developers	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution at an enterprise for Siebel applications.
Siebel System Administrators	Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.
System Integrators	Persons responsible for analyzing a business situation or using the analysis of a business analyst to build the integration solution at an enterprise for specific applications and or to develop custom solutions.
Siebel Alliance Partners	Partners developing applications leveraging the Siebel eBusiness platform.

Additional Documentation

In addition to the *Siebel Mobile Connector Guide*, there is an additional documentation set for Siebel eBusiness Applications provided on the *Siebel Bookshelf*. For general information about Siebel product documentation, see the *Siebel Bookshelf* home page.

The following documentation may be of particular benefit to you:

- *Siebel Tools Online Help*. This resource includes a reference to XML commands for the Siebel Web Engine.
- *XML Reference: Siebel eBusiness Application Integration Volume V*. This guide provides details for using XML to create integrations with Siebel applications.
- *Siebel Business Process Designer Administration Guide.* Read together with *Siebel Tools Online Help*, this guide can help you to gain an understanding of the Siebel eBusiness Application tools required for creating integrations.

Siebel Systems, Inc., reserves the right to modify the documentation for Siebel eBusiness Applications at any time. For updates to Siebel documentation, go to http://ebusiness.siebel.com/supportweb/.

If you want to order additional Siebel documentation and copies of the *Siebel Bookshelf* CD-ROM, go to Books Online at http://ebusiness.siebel.com/booksonline.

To access both SupportWeb and Books Online, you will need to provide the user name and password you received from Siebel Support Services (support@siebel.com).

Installing and Configuring Siebel Mobile Connector 2

Siebel Mobile Connector is a separately-licensed Siebel product option that is integrated into the architecture of Siebel 7.5, sharing the same Siebel Server, tool set (Siebel Tools), and installer as the rest of the Siebel eBusiness Applications suite. As such, this document focuses on the incremental steps required to enable Siebel Mobile Connector as part of installing Siebel eBusiness Applications.

Siebel Mobile Connector supports the same platforms supported by other Siebel eBusiness applications. Operating systems, databases, and browsers supported by other Siebel eBusiness applications are supported by Siebel Mobile Connector.

Installation Prerequisites

Before you begin installing Siebel Mobile Connector, make sure that you have the appropriate license keys for the number of users that your enterprise plans to support. You must enter the license key for Siebel Mobile Connector in order to use the product. The Siebel Mobile Connector components are installed automatically during the installation of Siebel Application Server and Siebel Web Engine.

For more information, see the *Siebel Server Installation Guide* for the operating system you are using.

Required Siebel Components

Siebel Mobile Connector requires the installation of the following components:

- **Siebel Server 7.5.** This component must include the Gateway Name Server and Siebel Web Engine.
- **Siebel eBusiness Application Integration.** This component is necessary for the Alert Business Service to invoke an appropriate outbound transport method.
- **Siebel Business Process Management.** This component is necessary for the Alert Business Service and provides access to the Business Process Administration component.

Optional Siebel Components

Installation of the following components is optional:

- **Siebel Tools.** This component is necessary for configuring the Siebel Mobile Connector views.
- **COM Data Control.** This component must be installed on your application server if your application will use this method for accessing the Siebel XML Web Interface. Also, it must be installed on the machine where you are running the Reference Configuration Sample application. This control should be automatically installed when Siebel eAI is installed. You can verify that the control is installed by viewing the registry and checking to see whether SiebelDataControl.SiebelDataControl.1 is registered.

- **Java Data Bean Interface.** This component must be installed on your application server if your application will use this method for accessing the Siebel XML Web Interface. This control should be automatically installed when Siebel eAI is installed. For more information on the Java Data Bean Interface, see the Siebel JAR files included with your installation: SiebelJI_common.jar and SiebelJI_ < lang > located in sea700\siebsrvr\CLASSES.

Configuring Server Components

During the installation process for Siebel Server, you must enable the Siebel Sales component group. As shown in Figure 2, you select this option in the Enable Component Groups dialog box that appears during the Siebel Server configuration process. If you do not enable this component group, you will not be able to use Siebel Mobile Connector.

NOTE: If you have already installed Siebel Server, you may enable these components through Siebel Sales or any other Siebel eBusiness application with Server Administration. For information, see "After Installation of Siebel Server" on page 28

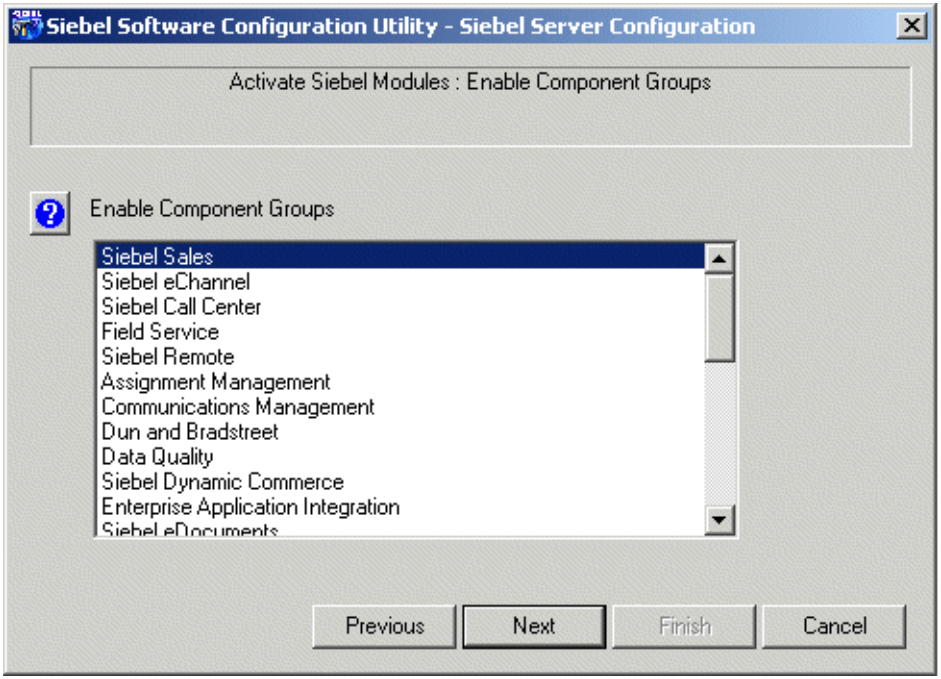


Figure 2. Siebel Sales Component Group

Optional Server Configuration Settings

During the installation process for Siebel Server, you can also enable the Workflow Management component group and the eBusiness Application Integration component to enable the Alert Business Service. As shown in Figure 4 and Figure 4, you select these options in the Enable Component Groups dialog box that appears during the Siebel Server configuration process. If you do not enable these component groups, you will not be able to send and receive alerts.

NOTE: If you have already installed Siebel Server, you may enable this component through Siebel Sales or any other Siebel eBusiness application with Server Administration. For information, see "After Installation of Siebel Server" on page 28

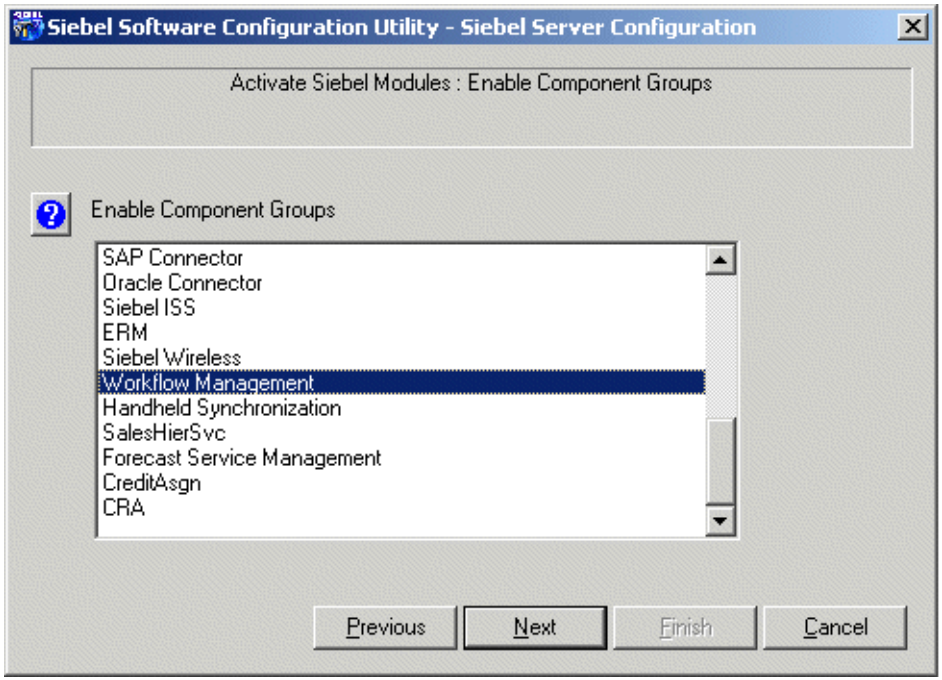


Figure 3. Workflow Management Component Group

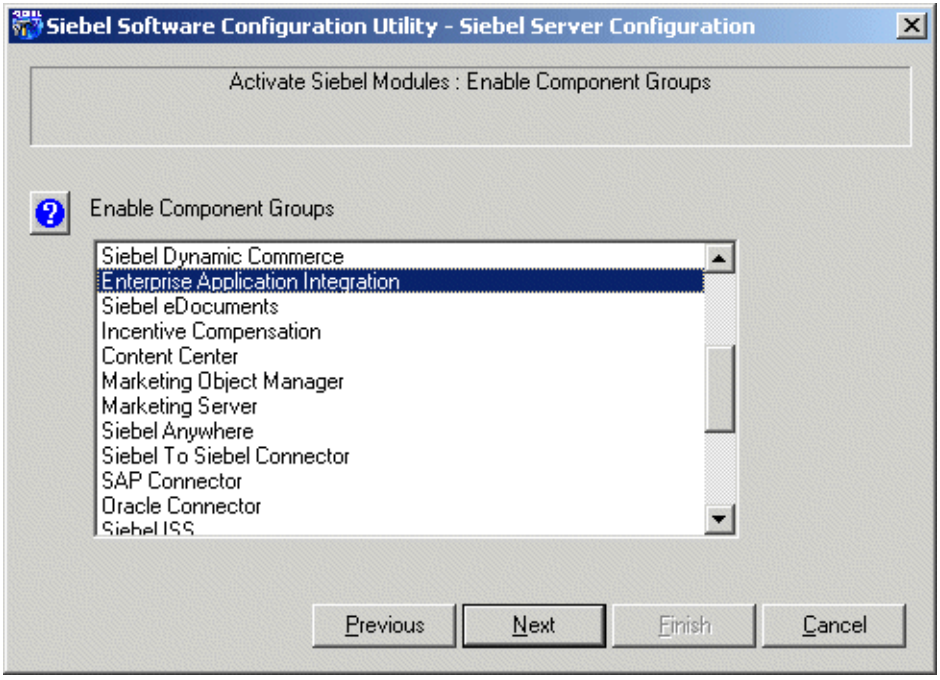


Figure 4. Enterprise Application Integration Component Group

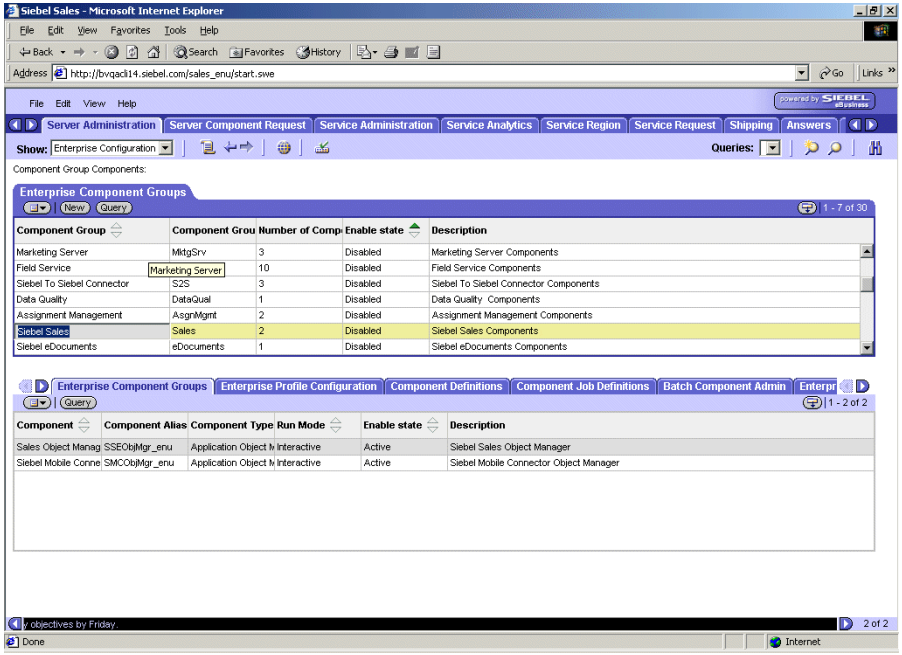
After Installation of Siebel Server

If you have already installed Siebel Server, you may enable the Workflow Management component and Siebel Sales component group through Siebel Sales or any other Siebel eBusiness application with Server Administration.

To configure the Siebel Server settings

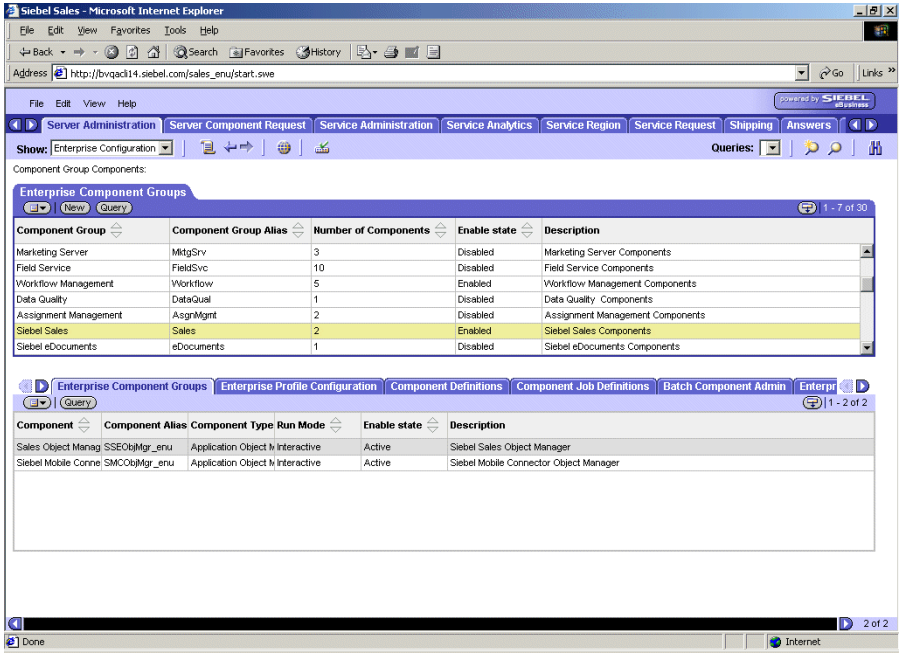
1 From the application-level menu, choose View > Site Map > Server Administration > Enterprise Configuration.

2 Query for Component Group = “Siebel Sales”.



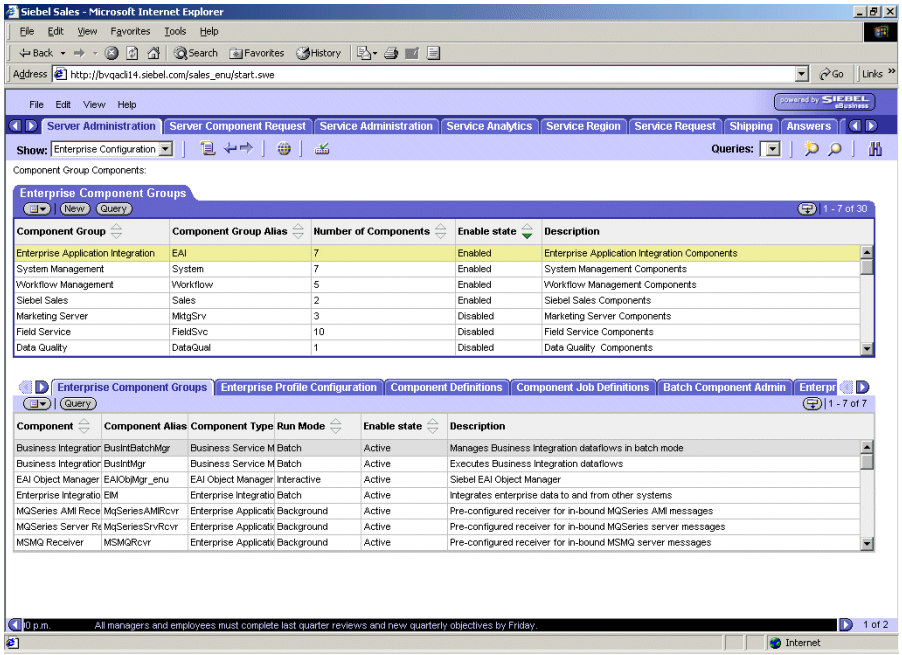
3 From the drop-down menu, select “Enable Component Group”.

The value of the Enable State column is “Enabled”.

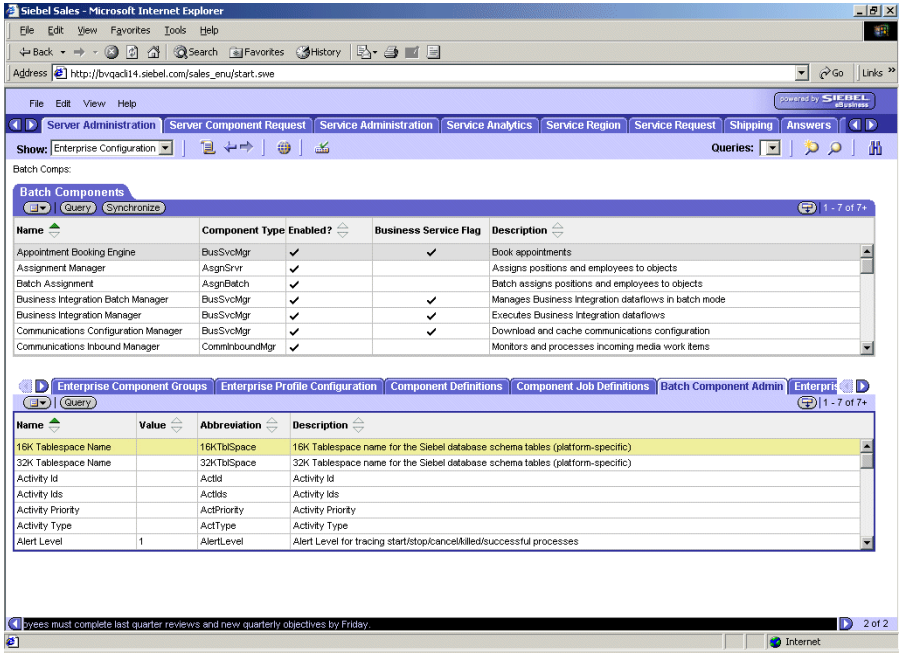


4 Enable the following Server Components: “Enterprise Application Integration” and “Workflow Management”

NOTE: This step is optional. It is only necessary if you want to enable the Alert Business Service.



5 In the lower applet, select the Batch Comp Admin tab and click the Synchronize button.



NOTE: Synchronization make take several minutes to be completed.

6 Restart Siebel Server.

Configuring Siebel Mobile Connector

This section describes how to modify the server configuration files to enable Siebel Mobile Connector to work in your environment.

Server Configuration Files and DLL Files

Various server configuration files and DLL files are created during the installation of Siebel Mobile Connector. These files are located in the <drive>:\<dir_name>\siebsrvr\bin\enu directory (where <drive> is the drive, and <dir_name> is the directory where Siebel Server was installed):

- "smc.cfg" for Mobile Connector
- "smcalert.cfg" for Alert Business Service

Two DLL files are created for Mobile Connector during installation. These files are located in the <drive>:\<dir_name>\siebsrvr\bin directory (where <drive> is the drive and <dir_name> is the directory where Siebel Server was installed):

- "sscalt.dll" ("sscalt.so" on UNIX)
- "ssmdbldr.dll" ("ssmdbldr.so" on UNIX)

NOTE: On UNIX systems, file locations will be different. In a typical install, the smc.cfg and smcalert.cfg files are located in a directory such as the following example: /15048/siebsrvr/bin/enu. The files sscalt.so and ssmdbldr.so are located in a directory such as the following example: /15048/siebsrvr/lib.

Configuring the smc.cfg File

The smc.cfg file contains parameters that may be configured before using the Siebel Mobile Connector application definition. However, it is not necessary and not recommended to change any parameters. Many of the parameter values contained in this file are read from the Gateway Server configuration files and do not need modification.

To set the interactivity mode

1. Go to <drive>:\<install_dir>\siebsrvr\BIN\ENU

   where <drive> is the drive where Siebel Server is installed and <install_dir> is the directory where you installed Siebel Server.

2. Open the smc.cfg file in a text editor such as Notepad.
3. Locate the section with the [SWE] parameters.
4. Set `HighInteractivity` to FALSE for standard interactivity mode or TRUE for high interactivity mode.
5. Save the file. If you are finished with configuration, then re-start the Siebel Server.

You can set the number of list rows returned by a query to a value other than the default (7 records). While the default is acceptable for a typical mobile application, your application could have special needs where it would be desirable to set a different value for the default. For example, if enabling a wireless browser application on a phone, you may only want 4 records to display on the small screen size. Or, if your application will be extracting dynamic grammars, you may want to retrieve a larger number of records when doing this batch process (for instance, 100 records per query).

To set the number of list rows returned by a query

1. Go to <drive>:\<install_dir>\siebsrvr\BIN\ENU

   where <drive> is the drive where Siebel Server is installed and <install_dir> is the directory where you installed Siebel Server.

2. Open the smc.cfg file in a text editor such as Notepad.
3. Locate the section with the `NumberOfListRows` parameters.

   By default, the value is 7. Change the value of this parameter if you want to specify a different number of rows.

4 Save the file. If you are finished with configuration, then re-start the Siebel Server.

NOTE: For information about changing parameter values in the smc.cfg file that are defined during the configuration of Siebel Server or Gateway Server, see the *Siebel Server Administration Guide* in the *Siebel Bookshelf*.

Configuring the smcalert.cfg File

The smcalert.cfg file should be configured before using the Alert Business Service. This file specifies the transport mechanism to be used by the Alert Business Service. If a transport mechanism is specified here, it is used by default for all alerts created by Siebel Mobile Connector. However, it is possible to specify a transport mechanism in the workflow for an alert, overriding the settings in the smcalert.cfg file. For instructions, see the "Configuring Alerts" on page 122.

The transport mechanisms allow the transportation of messages between another system the Siebel eBusiness Application Integration (eAI) environment. Alert Business Service supports all the transport mechanisms available within the eAI, including MQSeries, MSMQ, HTTP, Java Data Beans, SAP IDOC, SAP BAPI and others.

An example of a file configuration is as follows:

```
[EAI MSMQ Transport]

MsmqPhysicalQueueName=fromsiebel

MsmqQueueMachineName=machine1701


[SMC Alert]

WorkflowDelete = SMCAlert-Delete.xml

WorkflowInsert = SMCAlert-Insert.xml

WorkflowOlfValue = SMCAlert-OldFieldValue.xml

WorkflowNewValue = SMCAlert-NewFieldValue.xml
```

To set the default transport mechanisms

1. Go to <drive>:\<install_dir>\siebsrvr\BIN\ENU

   where <drive> is the drive where Siebel Server is installed and <install_dir> is the directory where you installed Siebel Server.

2. Open the smcalert.cfg file in a text editor such as Notepad.

3. Add a parameter to the file for each transport mechanism you want to use for the Alert Business Service. For each added transport mechanism, it is also necessary to add the required parameters used for configuring it.

   The MSMQ transport mechanism is displayed by default. To configure MSMQ for use with the Alert Business Service, enter the name of the MSMQ Queue for `MsmqPhysicalQueueName` and enter the machine that owns the queue specified by the physical queue name for `MsmqQueueMachineName`. You can also set any optional parameters that you want to configure.

4. Save the file. If you are finished with configuration, then re-start the Siebel Server.

NOTE: You can also use the Reference Configuration Sample to configure the smcalert.cfg file. For more information, see "SMC Alert Welcome Screen" on page 109.

For information on supported transport mechanisms and the parameters for each, see the eAI documentation in the *Siebel Bookshelf*, especially the EAI Transports and Interfaces Overview.

Configuring a User Agent for Siebel Mobile Connector Applications

The XML Web Interface requires that a user agent be configured to identify the application. A user agent could be a Web browser or a third party application. The user agent is denoted in the HTTP header information in XML documents sent to the XML Web Interface when third party applications send requests to SWE.

Using the Web Browser Administration screen, you may set up the user agent and its capabilities within the Siebel application. This screen is not available within the Siebel Mobile Connector application, but within a core Siebel eBusiness application such as Siebel Sales. Web browser capabilities identify what an end-user's browser or application can and cannot do within the Siebel Web Engine.

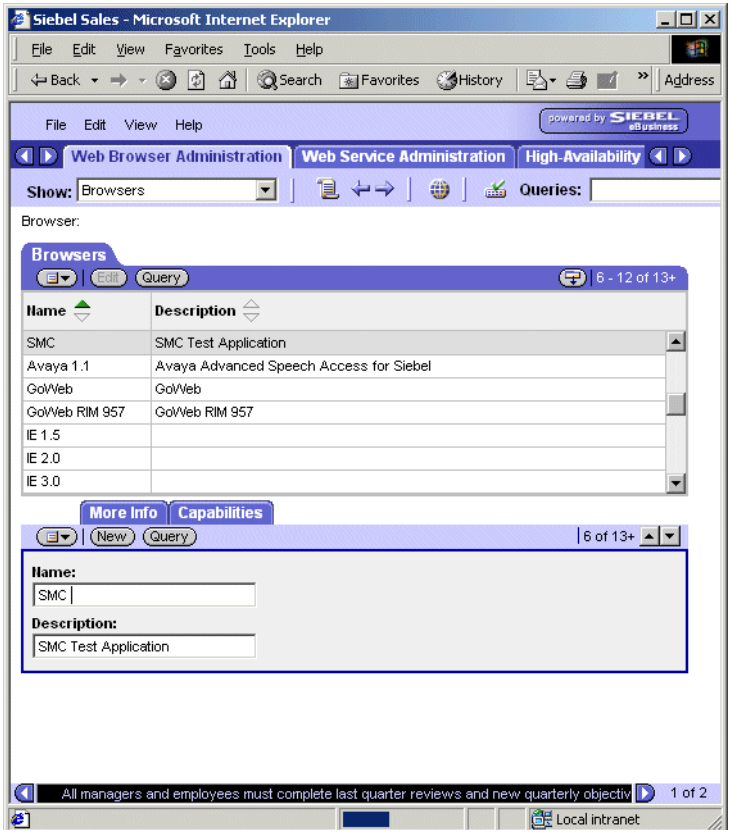
The Siebel Mobile Connector requires that you set the capability “Voice Application” or “Mobile Application” of the browser you want to use to TRUE. This is necessary to call the GetSMCUpdate method.

To set the VoiceApplication capability to TRUE

1. From the application-level menu, choose View > Site Map > Web Browser Administration > Browsers.

2 On the Browsers screen, query for or add the browser to be used by the application.

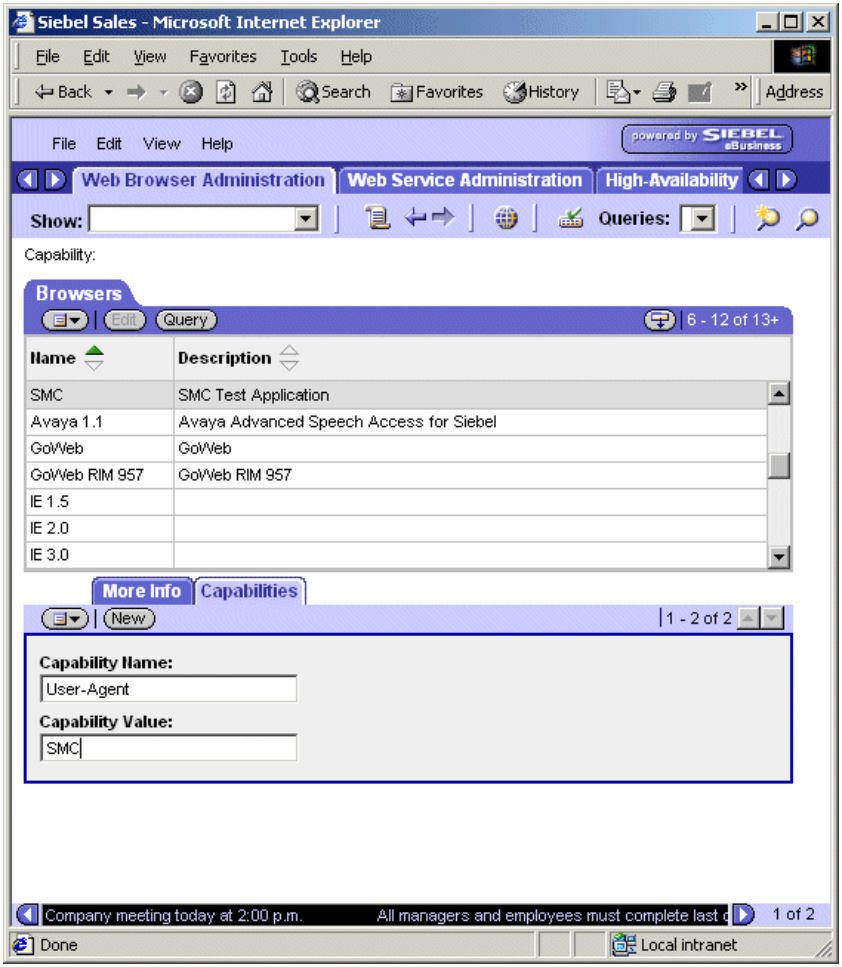
For example, query for the Web browser version that your client application will be running (for example, enter IE 5.5) and select the Web browser version. If you want to add a browser instead of querying for a listed browser, click the New button and enter the browser's name and description.



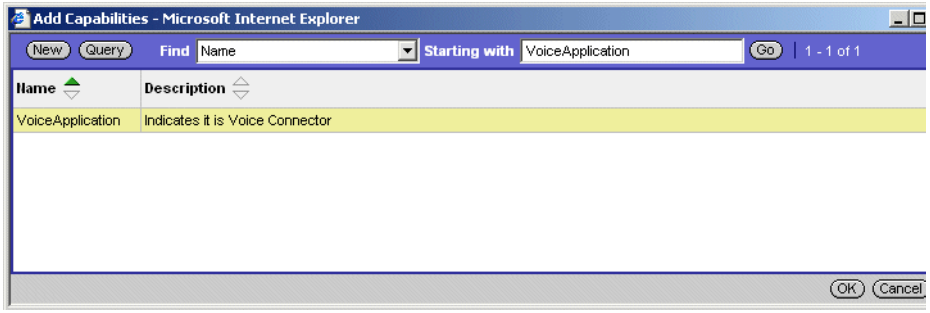
3 Click the Capabilities tab in the lower applet.

4 Within the Capabilities Applet, click New record.

The pick list for the Capability Names appears.



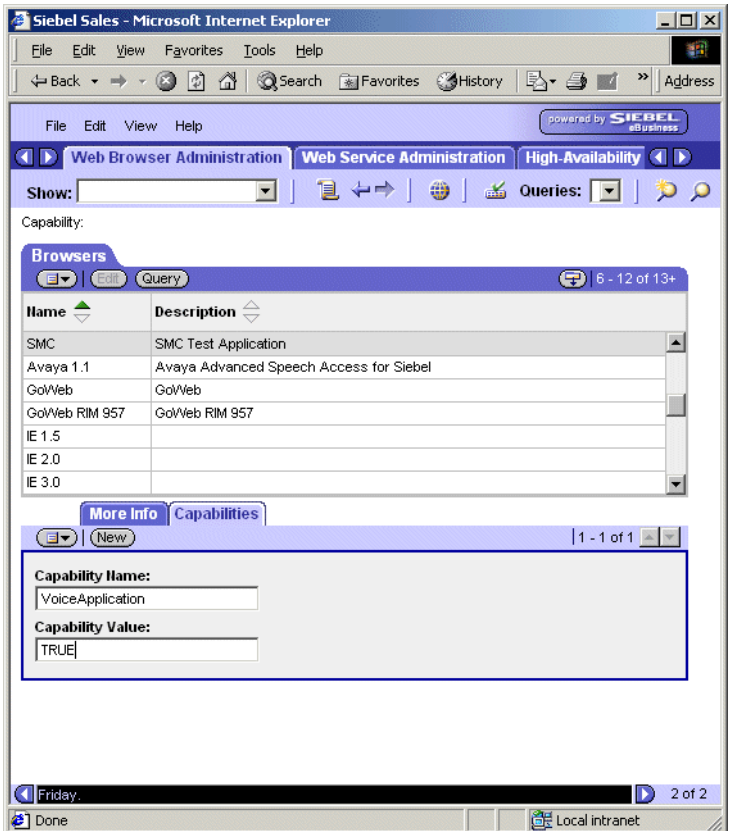
5 In the Capability Name search field, enter “VoiceApplication” and click Go.



6 Upon retrieving “VoiceApplication,” click Ok.

NOTE: Although this may seem to indicate only voice applications may use this capability, this is not the case. The system does not check to see what type of application is actually connecting only that its browser type and this capability have been registered.

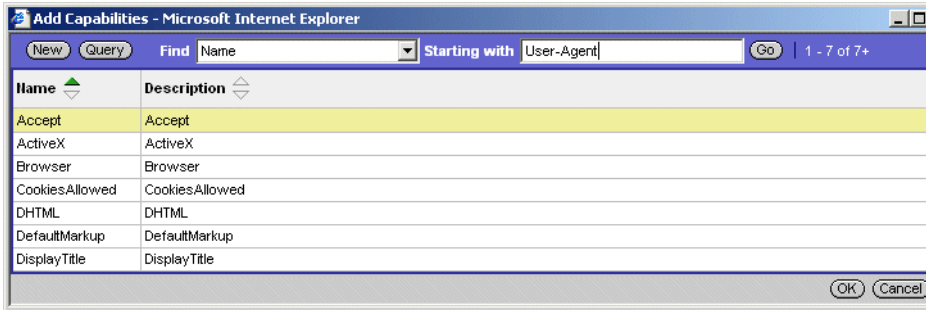
7 Click the Edit button to edit the Capability Value field, and then enter “TRUE.”



8 Save the record.

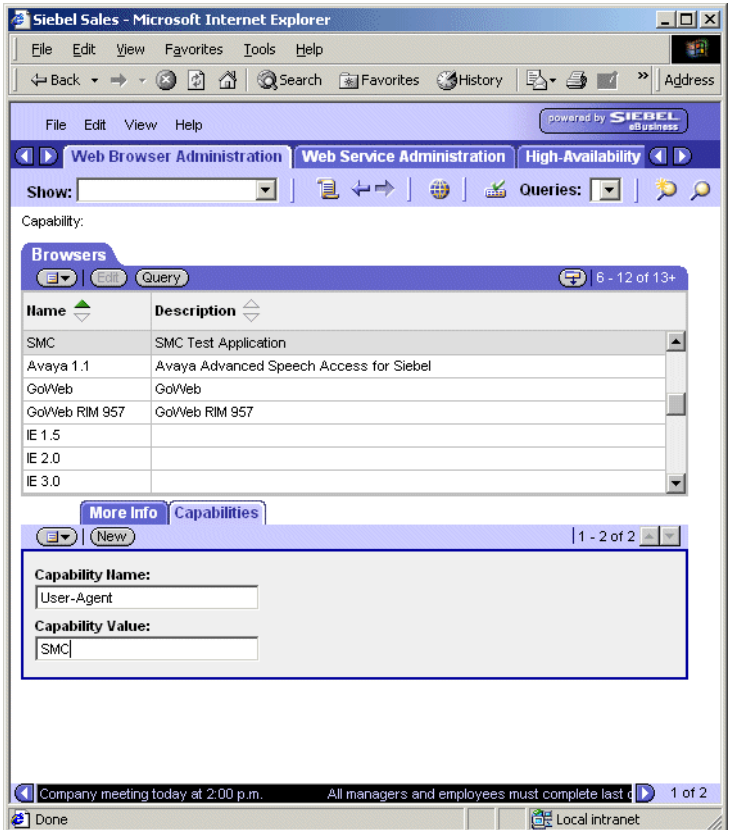
By stepping off the record, an implicit save will occur, or select Save from the dropdown menu to explicitly save the record.

9 Click the New button to add the User-Agent Capability (refer to steps above).



10 Click the Edit button to add the User-Agent capability value, and then save the record.

The value that you input here must be used in the header of the XML query that will be passed to the Siebel Web Engine.



11 Restart the Siebel Server.

Configuring the Application Definition

Siebel application definitions are comprised of screens, which include various views. Views include list views, detailed views, parent views and child views. In Siebel Mobile Connector, there are ten pre-configured screens that provide the most frequently and commonly used functions for mobile applications. Based on user responsibility type, a user can see limited or all of the screen when logged in through user authentication.

For a reference to the application definition, see Appendix , “Siebel Mobile Connector Application Definition Quick Reference”.

Customizing the Application Definition

The Metadata Business Service and Reference Configuration Sample give you a way to alter the application definition through XSL style sheets without permanently changing the application definition. If you want to customize the application definition, you can use Siebel Tools. You can modify the code of underlying object definitions to change the look and feel of an application.

For more information, see the *Siebel Tools Reference* in the *Siebel Bookshelf*.

Setting User Responsibilities

As with all Siebel applications, access to views is determined by a user's responsibilities. Siebel Mobile Connector offers six user responsibility types. Administrators must assign user responsibilities from these user types.

User Type	Responsibilities
Anonymous User - SMC	Access to: ■ SMC Branch Locator View ■ SMC eService Order View ■ SMC eService Request View
Call Center Representative - SMC	Access to all views in: ■ SMC Account Screen ■ SMC Activity Screen ■ SMC Contact Screen ■ SMC Employee Screen ■ SMC Opportunity Screen ■ SMC Service Request Screen
Field Service Representative - SMC	Access to all views in: ■ SMC Account Screen ■ SMC Activity Screen ■ SMC Contact Screen ■ SMC Employee Screen ■ SMC Service Request Screen
Registered Customer - SMC	Access to: ■ SMC Branch Locator View ■ SMC eService Order View ■ SMC eService Request View

User Type	Responsibilities
SMC Administrator	Access to: ■ SMC Responsibility View
Sales Representative - SMC	Access to all the views in: ■ SMC Account Screen ■ SMC Activity Screen ■ SMC Contact Screen ■ SMC Opportunity Screen ■ SMC Employee Screen

Installing and Configuring the Sample Application

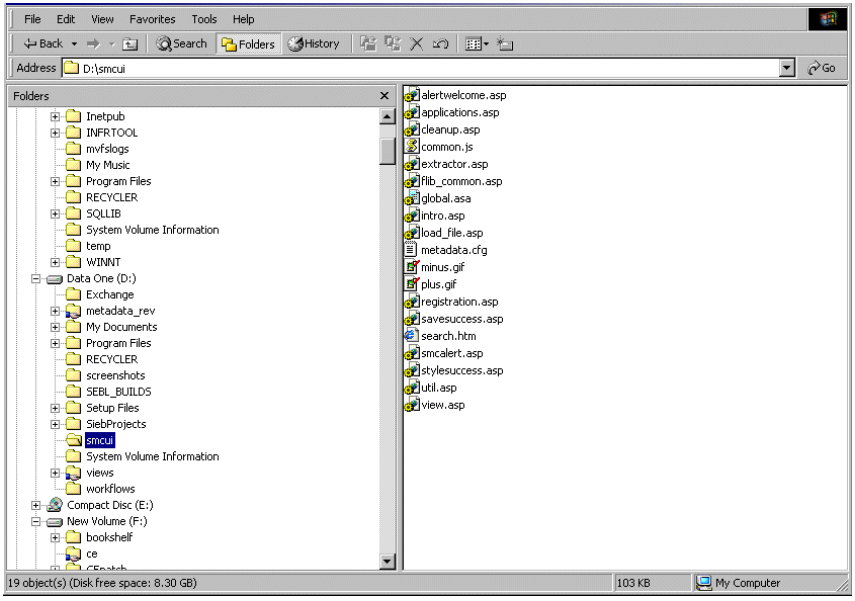
The Reference Configuration Sample is an application that allows you to access the Siebel Mobile Connector Metadata Business Service API through a graphical user interface. You can use the Reference Configuration Sample to generate style sheets and alerts. It can also be used to configure alerts for a transport method. The sample application is not shipped with Siebel 7.5 but is available for download through the Siebel Support Web.

The sample application must be installed on a machine running Microsoft Internet Information Server. Also, as noted previously you must have the Siebel COM Data Control control installed and configured on the machine where the sample application is installed.

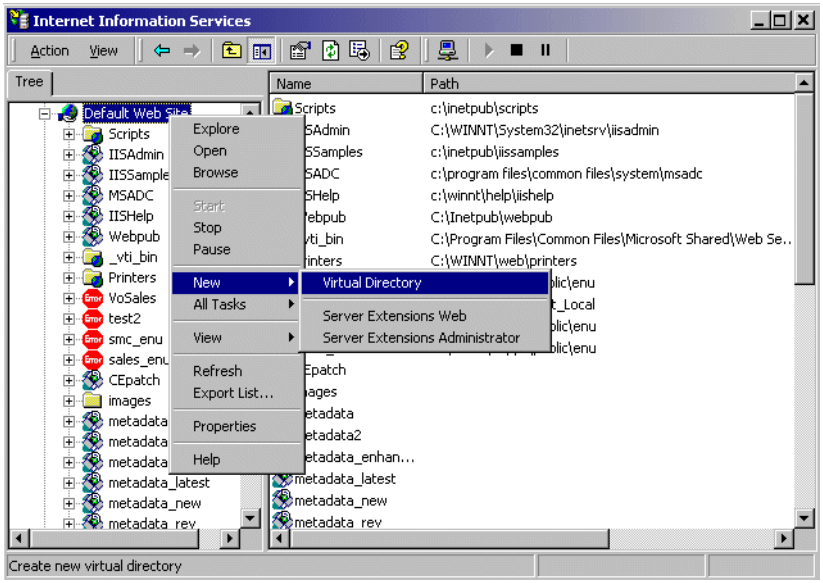
You can follow the instructions below to install and configure the sample application.

To install and configure the sample application

1 Download the self-extracting ZIP file, smcrefconfig.exe, from Siebel Support Web and unzip the contents to an appropriate Web directory on a machine running Microsoft Internet Information Server (IIS).

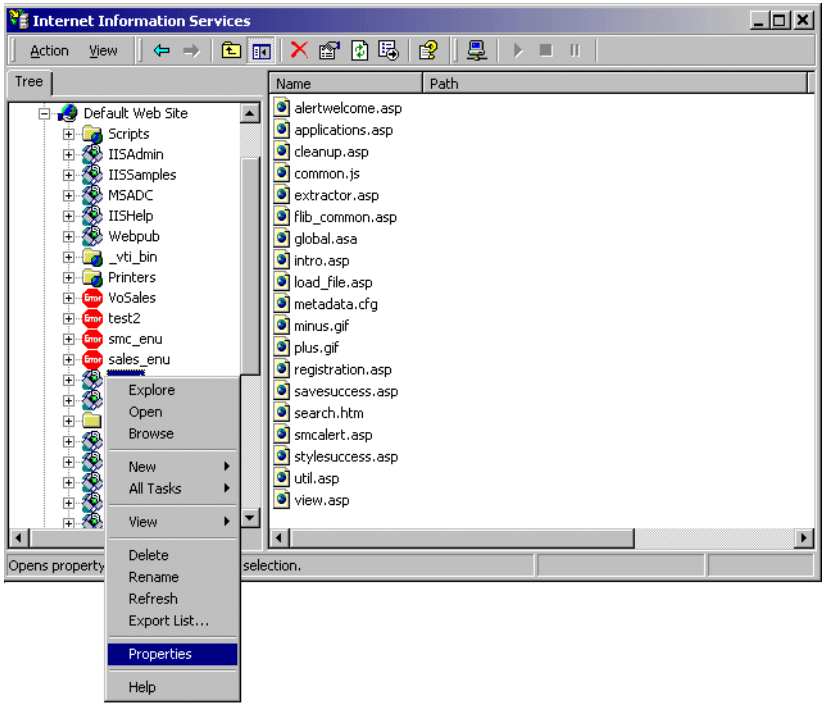


2 Create a virtual directory on IIS and point it to the Web directory from the previous step.

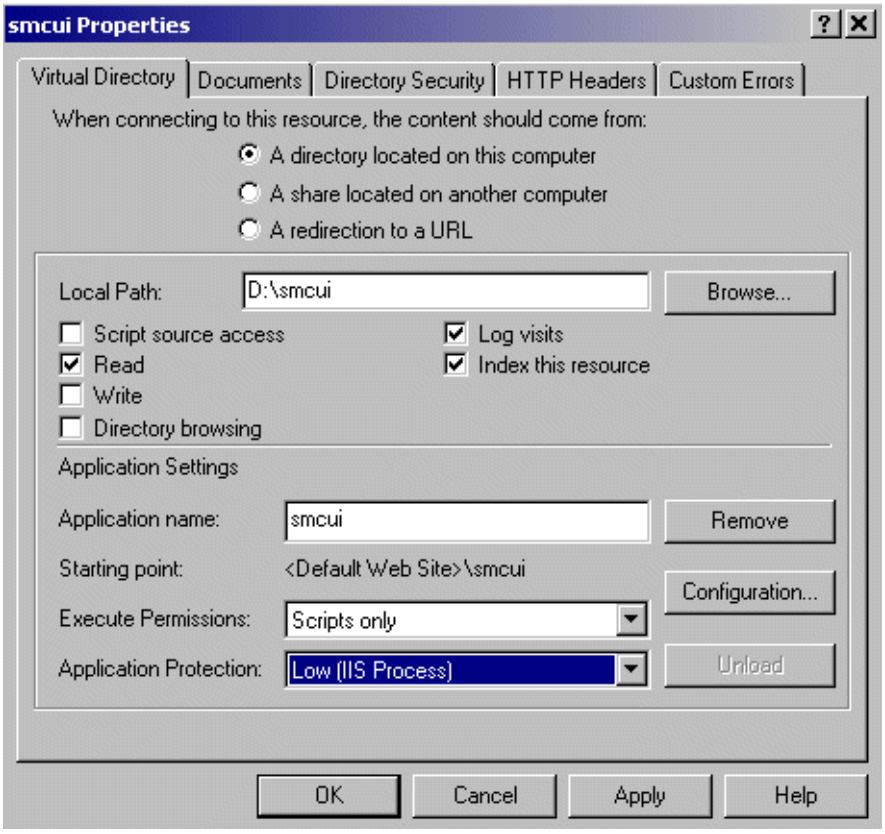


NOTE: For detailed instructions regarding this step, consult the Microsoft Internet Information Server documentation.

3 Right-click on the virtual directory (it is located under the Default Web Site) and select Properties.

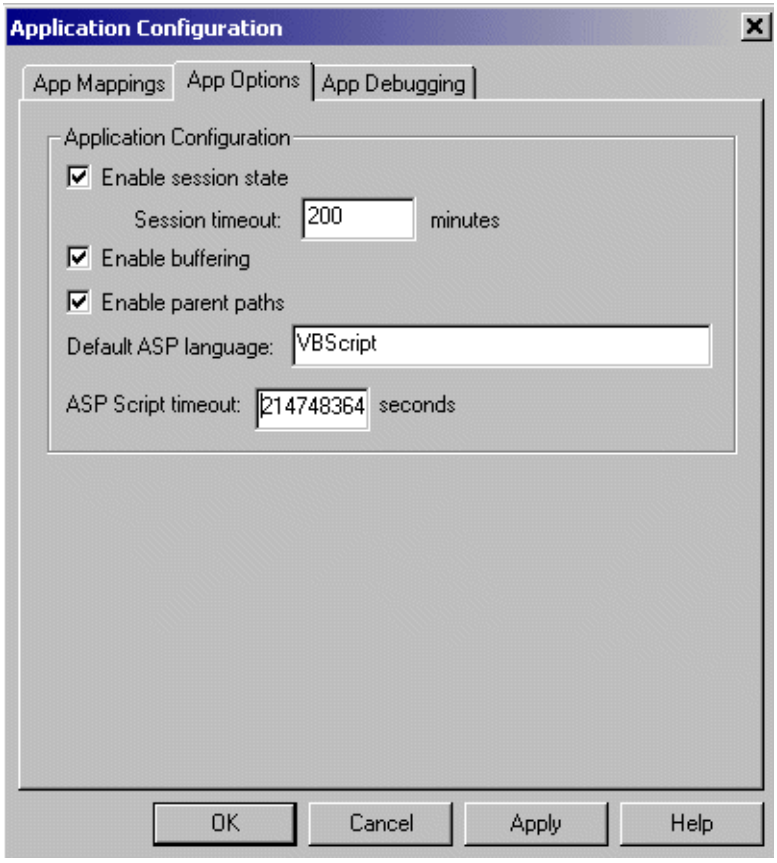


4 Verify that Execute Permissions is set to Scripts Only and that Application Protection is set to Low (IIS Process).



5 Click the Configuration button.

A dialog box appears. On the dialog box, click App Options. On the App Options tab, set the ASP Script time-out value to a large number (for example, 2147483646 seconds) and set the session time out to a large number (for example, to 200 minutes).



NOTE: The process of creating a new subscription for a typical new application can take 10 minutes or longer. If the application is very large, the process could take a few hours. To avoid page time-out errors, you need to set the script and session time-outs to large values.

6 Click OK.

The Directory window appears.

7 From the Directory window, select the Directory Security tab. In the Anonymous access and authentication control section, click the Edit button.

The Anonymous User Account dialog box appears.

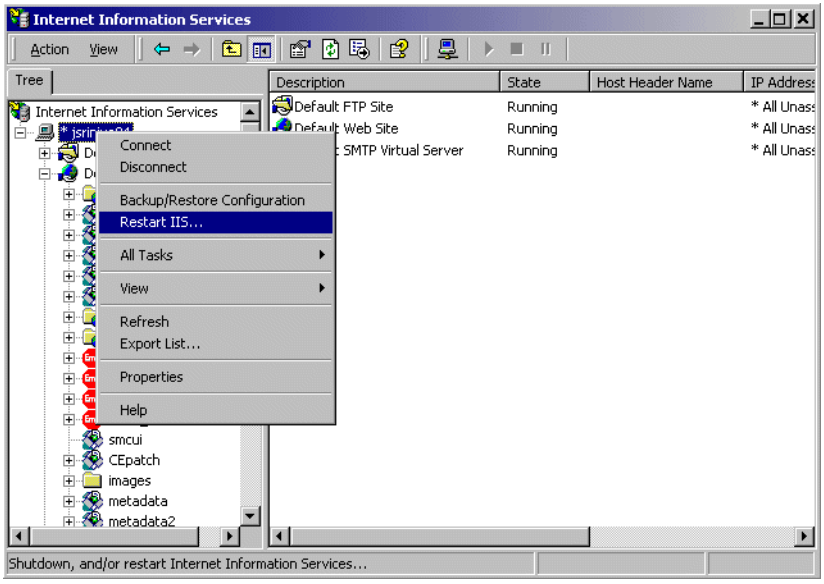
8 Verify that the user listed in this dialog box has read/write permissions to the IIS directory. If necessary, change the user to a user who has read/write permissions for the IIS directory (such as your Windows login account name).

9 Edit the following parameters in the metadata.cfg file:

```
GatewayServer=<Siebel Gateway Server name>

EnterpriseServer=siebel

Port=<port number>

SiebelServer=<Siebel Server name>
```

```
Language=ENU
```

10 Restart IIS.



11 Open a Web browser and type the following in the Address bar:

```
http://<machine name>/<virtual_directory_name>/registration.asp
```

The Reference Configuration Sample application is displayed.

Working with Siebel Mobile Connector 3

This chapter provides an overview of data transfer mechanisms and information about working specifically with real-time access to data and data updates. Topics include a review of the format of XML user data and commonly used XML commands, updating and synchronizing data, and adding support for multiple languages.

Overview of Data Transfer Mechanisms

Siebel Mobile Connector provides three mechanisms for getting data to and retrieving data from partner applications and the Siebel application server: real-time access, data updates, and pushed alerts.

- **Real-time access to data.** You can add, modify and delete records in real-time using XML commands. User data is returned as XML documents. This capability is useful for accessing contact details, updating opportunities, accessing contacts, and so forth.
- **Data updates.** You can query the Siebel database for new information entered for a specific user. This capability is useful for retrieving information that has changed or been added since the last time a user has logged into the application, such as new activities or new opportunities.
- **Pushed alerts.** You can also obtain pushed alerts from the Alert Business Service. The Alert Business Service creates an XML document with the relevant, pre-defined data and sends it to the third-party application. For example, two service technicians could be notified that a service request has been reassigned from one to the other.

NOTE: If you are using the COM Data Control or Java Data Bean to send HTTP or XML requests to SWE, then the XML output received from SWE will contain an encoding property of UTF-16. In other words, the XML header will look like this: <?xml version="1.0" encoding="UTF-16"?>. If the XML output must be loaded in a web browser or an XML parser, then you must remove this property or set it to UTF-8. The header will like this: <?xml version="1.0" ?> or <?xml version="1.0" encoding="UTF-8"?>. For the latest information on UTF encoding, see Siebel Support Web.

Real-Time Access to Data

This section provides an overview of working with Siebel Mobile Connector to obtain real-time access to Siebel data from third-party applications. Mobile applications make requests to Siebel Mobile Connector through HyperText Transfer Protocol (HTTP), Siebel COM Data Control, Siebel Java Data Bean interface or any other Siebel object interface that supports the GetService method.

See the *Siebel Tools Online Help* for information on object interfaces including Siebel COM Data Control and the Siebel Java Data Bean interface.

NOTE: Examples in this document show commands in the HTTP format. In the examples, spaces in the HTTP requests are shown replaced by a " + ". It is recommended that you replace the spaces in HTTP requests by " + " symbols.

XML Commands and SWE Methods

You can add, modify and delete records in real-time using XML commands. The following are some common XML commands that can be issued from third-party applications to Siebel Mobile Connector which in turn uses the XML Web Interface provided by SWE.

- CanInvokeMethod
- ExecuteLogin
- ExecuteNamedQuery
- GotoPage
- GotoPageTab
- GotoView
- InvokeMethod
- LoadService
- Login
- Logoff
- ReloadCT

The following are some methods commonly used with SWE:

- CollapseTreeItem
- CopyRecord
- CreateRecord
- DeleteQuery
- DeleteRecord
- Drilldown
- EditRecord
- ExecuteQuery
- ExpandTreeItem
- GetSMCUpdate
- GotoFirstSet
- GotoLastSet
- GotoNextSet
- GotoPreviousSet
- GotoView
- MoveDown
- MoveUp
- NewQuery
- NewRecord
- NextTreeItem
- PickNone
- PickRecord
- PositionOnRow
- PreviousTreeItem
- RefineQuery
- SaveQueryAs
- SelectTreeItem
- SortAscending
- SortDescending
- ToggleTo
- UndoRecord
- WriteRecord

NOTE: SWE expects the correct spelling and valid values for all commands, methods and arguments. Invalid SWE commands, methods and arguments are ignored. No error message is returned by SWE for such errors and the calling application may experience unexpected results.

See the *Siebel Tools Online Help* for information on the XML interface to SWE.

XML User Data

This section gives a summary of the most common XML tags returned to the third-party application following a request for user data to SWE. The user data is returned as an XML document.

In the Siebel database, data is contained in applets, which are contained within views, and views in turn are contained in screens. To access data in the Siebel database, make an XML request to the SWE. If you want to retrieve only data, set the parameter SWEDataOnly to TRUE. By using this flag, you can ensure the XML document contains only data tags and does not contain any user interface navigation elements such as drop-down menus, page tabs, and so on.

XML Page Content

In order to support different implementations, Siebel Mobile Connector defines user interfaces and user data in the XML format.

In response to the XML requests of third-party applications, Siebel Mobile Connector returns XML pages as output. The XML page output is based on the application definitions, including application, screen, view, applet and controls, that are defined in a repository. The output can be tailored to the application by including only data specified during configuration by applying XSL style sheets generated by Siebel Mobile Connector for this purpose.

Section	Required/Optional	Description
XML version and encoding	Required	Describes the version of XML supported and the type of encoding used. Appended in all XML pages.
Application	Required	Describes the application name, such as Siebel Mobile Connector or Siebel Sales Enterprise, that the third-party application is connected to and interacting with. Appended in all XML pages.
User agent markup	Required	Describes the default markup language that is supported. It is based on the user-agent in the HTTP request header.

Section	Required/Optional	Description
Navigation elements	Optional	Contains the following user interface information (the UI is defined in Siebel Tools): Menu, Tool bar, Screen bar, Thread bar, Page item. The information under this tag can be turned off by specifying SWEDataOnly = TRUE in the HTTP request.
Form definitions	Optional	Contains the user interface for pre-defined queries (the UI is defined in Siebel Tools). Like the navigation elements, this information is generated by default. The information under this tag can be turned off by specifying SWEDataOnly = TRUE in the HTTP request.
Active Screen, View and Applets definition and User Data	Optional	Contains the current active screen and view information, applets, and the record (user data) defined in that view. Note that this section contains some UI elements that will not be included in data only mode. This section is generated by default. To return only this information, specify SWEDataOnly = TRUE in the HTTP request.

Common XML Tags

The following table provides a list of common XML tags returned in user XML documents and their attributes.

Table 1. Common XML Tags and Attributes

XML Element	Description
APPLICATION	Specifies the name of the application. For example, < APPLICATION NAME = "Siebel Mobile Connector" > where APPLICATION is the tag, NAME is its attribute and Siebel Mobile Connector is the value of NAME attribute.
USER_AGENT	Specifies information about the user agent or the browser type that made the XML or HTML request.
SCREEN	Specifies information about the name and title for the currently active screen. Contained inside the APPLICATION element.
VIEW	Describes the name and title of the currently active view (similar to the SCREEN element).
APPLET	Returns additional information in addition to the name and title of the applet. Contained inside the VIEW element.
MODE	Describes what mode the applet is in. Possible values include EDIT or BASE. EDIT specifies that the applet allows modification, deletion, creation and querying of records. BASE specifies that the applet is read only and cannot be modified.
NO_INSERT, NO_MERGE, NO_DELETE, NO_UPDATE, NO_EXEC_QUERY	Provide a filter to what specific edit mode operations are possible for the applet. If any of these attributes are TRUE, then that particular operation is not possible. For example, if NO_INSERT attribute is TRUE then new records cannot be inserted into the applet.The third-party application can customize the associated commands based on these attributes. For example, if the NO_EXEC_QUERY attribute is set to FALSE for an applet, this indicates that the third-party application should be able to query for a contact using that applet.
CLASS	Specifies the C + + class the applet belongs to. For example, in the first sample in this section, the CLASS attribute has a value of CSSFrameListBase, which means it is a List applet. The second sample has a CLASS value of CSSFrameBase, which means it is a Form applet.

Table 1. Common XML Tags and Attributes

XML Element	Description
ROW_COUNTER	Gives an indication of the number of data records returned. A " + " at the end indicates that there are more records than that returned.
RS_HEADER	Contains the COLUMN element.
COLUMN	Specifies the column details for the data records.
NAME, DISPLAY_NAME, TEXT_LENGTH	Specify information about the name, title and text length of the columns respectively.
DATATYPE	Describes what kind of data type the column represents. For example, the phone number has a data type of “phone” and an email has a data type of “email”. This information could be used by the third-party application to make a call or send an email.
REQUIRED	Specifies whether or not the column is required. This information is useful when creating new records. The third-party application can determine what field information is mandatory by looking at this attribute.
FORMAT	Specifies the format of the data. For the Date data type this attribute should contain the acceptable Date Format (refer to the following sample). For revenue and other price related fields this attribute will have the format for the dollar amount. The third-party application can use this to get or display the right information back to the user.
CALCULATED	Specifies that the column has been calculated, for example, by using mathematical expressions. The column has not been directly derived from the database tables. This information could be useful during record creation.
FIELD	Specifies the name of the FIELD element in the business component that the column refers to. The FIELD element contains the actual data. The third-party application would make use of both FIELD and COLUMN elements to get more information on the data. FIELD is useful in determining what fields to query on while fetching a particular record.
READ_ONLY, LIST_EDITABLE	Specifies whether the column is editable or just read only. This information could be useful to the third-party application when modifying certain columns.

Table 1. Common XML Tags and Attributes

XML Element	Description
NUMBER_BASE, TEXT_BASED	Indicates whether the column/field is a number or text.
RS_DATA	Contains the XML tags that hold the actual data.
ROW	Identifies the row id of the data in the attribute ROW_ID. This information is very useful in querying for a particular row of data and getting the detailed information for that row.
SELECTED	Indicates that the particular row is selected on the user interface.

Retrieving Data Only

If SWEDataOnly is set to TRUE, all elements contained within both the NAVIGATION_ELEMENTS tag and the FORM tag will not be returned. For example, this code fragment represents an XML document where SWEDataOnly is set to FALSE:

```
<?xml version="1.0" encoding="UTF-8" ?>

- <APPLICATION NAME="Siebel Mobile Connector">

  <USER_AGENT MARKUP="HTML" />

+ <NAVIGATION_ELEMENTS>

+ <FORM ACTION="/smc/start.swe" METHOD="POST" NAME="SWEForm4">

+ <SCREEN CAPTION="Accounts" ACTIVE="TRUE" NAME="SMC Account
Screen">

  </APPLICATION>
```

In contrast, this code fragment represents an XML document where SWEDataOnly is set to TRUE:

```
<?xml version="1.0" encoding="UTF-8" ?>

- <APPLICATION NAME="Siebel Mobile Connector">

  <USER_AGENT MARKUP="HTML" />
```

```
+ <SCREEN CAPTION="Accounts" ACTIVE="TRUE" NAME="SMC Account
Screen">

  </APPLICATION>
```

Retrieving Detailed Information About the Data

The RS_HEADER section holds detailed information about each data column. For instance, if the third-party application wants to detect if a particular column holds a phone number then it should lookup the DATATYPE attribute in the COLUMN element (under the RS_HEADER section) and then get the data from the FIELD element. The FIELD attribute of the COLUMN element gives a link to the FIELD element, which holds the actual data.

NOTE: Using the field attributes in the RS_DATA section for data type detection is not recommended. The reason is that this information is not guaranteed to be a constant. The RS_DATA might change if the object definition, field names in this case, are changed in Siebel Tools.

Authenticating Users

Authentication is the process of verifying the identity of a user. Siebel Mobile Connector uses the Siebel security adapter authentication architecture for authenticating Siebel application users to external partner applications. This architecture supports authentication to Microsoft Active Directory Server and LDAP-compliant directories. Additionally, partners may also create their own authentication mechanism by writing to the Siebel Security Adapter API. For more information, see the *Security Adapter SDK* available on Siebel Support Web.

Partner applications must login to the Siebel Web Engine to instantiate a user session and must logout to terminate the session. A user's session is managed in SWE by using cookies or an authentication mode without cookies.

Logging In

Logging in to SWE is required to instantiate a new user session. The command ExecuteLogin is used to supply the user credentials and log in.

Shown below is an example of how to construct a login command by using an HTTP request:

To log in to SWE

- Send the ExecuteLogin command in a HTTP request to SWE, with a valid user name and password.

  Example (where your user name/password is WLEE):

  ```
  http://localhost/smc_enu/
  start.swe?SWECmd=ExecuteLogin&SWEUserName=WLEE&SWEPassword=WLEE&
  SWESetMarkup=XML
  ```

NOTE: SWESetMarkup is a required parameter. Generally, when using Siebel Mobile Connector you can set the SWESetMarkup parameter to HTML, WML, or XML. However, you may not use the GetSMCUpdate method or receive alerts in any other markup than XML.

Logging Off

Logging off of SWE is required to terminate a new user session. The command Logoff is used to log off.

To log off from SWE

- Send the Logoff command in a HTTP request to SWE.

  Example (where your user name/password is WLEE):

  ```
  http://localhost/smc_enu/
  start.swe?SWECmd=Logoff&SWESetMarkup=XML
  ```

NOTE: There is a default time-out set by the SWE engine if no user action has taken place.

Session Management

When a user logs in to the Siebel Web Engine, SWE dynamically generates session cookies or uses an authentication mode without cookies. Cookies are generated by default and include a `Session ID` that is used to track the session. If cookies are disabled or if a user's browser does not support cookies, then the `Session ID` for each page is included in its URL.

For more information, refer to the *Security Guide for Siebel eBusiness Applications* in the *Siebel Bookshelf*.

Retrieving Data

This section describes how to retrieve data from SWE by using the XML Web Interface. Included is an overview of navigating to a screen, navigating within a screen, queries and updates.

Navigating to a Screen

Navigating to a screen is required to retrieve data about the screen's views and applets. The command GotoPageTab is used to go to a specific screen.

To navigate to a screen

1. Login to SWE. For more information, see "Logging In" on page 65.
2. Navigate to the screen to which you want to go.

   Example (where you are navigating to the SMC Opportunity Screen):

   ```
   http://localhost/smc_enu/
   start.swe?SWECmd=GotoPageTab&SWEScreen=SMC+Opportunity+Screen&SW
   EDataOnly=TRUE&SWESetMarkup=XML
   ```

NOTE: GotoPageTab executes the default PDQ (pre-defined query) for that screen.

Following is a list of screens provided in the Siebel Mobile Connector application definition to which you can navigate.

Screen Name	Display Name
SMC Account Screen	Accounts
SMC Activity Screen	Activities
SMC Contact Screen	Contacts
SMC Opportunity Screen	Opportunities
SMC Service Request Screen	Service Requests
SMC eService Request Screen	eService Requests
SMC eService Order Screen	Service Orders
SMC Branch Locator Screen	Branch Locator

SMC Employee Screen	Employees
SMC Responsibility Screen	Responsibilities

Navigating within a Screen

Navigating within a screen is required to perform an action on data from a screen's views and applets. You can use to the GotoView command to go to a particular Siebel view, where you can access the applets available to that view. The GotoView command requires the name of the view to be passed in the SWEView parameter.

To navigate to a view or applet

1. Login to SWE and navigate to the screen to which you want to go. For more information, see "Logging In" on page 65 and "Navigating to a Screen" on page 67.
2. Navigate to the view and applet to which you want to go.

   Example (where you are navigating to the SMC Opportunity Detail - Contacts View):

   ```
   http://localhost/smc_enu/
   start.swe?SWECmd=GotoView&SWEView=SMC+Opportunity+Detail+-
   +Contacts+View&SWENeedContext=false&SWEBID=-
   1&SWEKeepContext=1&SWESetMarkup=XML
   ```

For a complete list of the view and applet names to which you can navigate, see Appendix A, "Siebel Mobile Connector Application Definition Reference."

Querying Items

To perform a query, you must navigate to the screen that allows queries. Then you must send two separate requests to SWE: first, you must execute the NewQuery command; second, you must execute the ExecuteQuery command. In the ExecuteQuery command block, you must specify a parameter to identify the column (the field you want to search) and a value to indicate the search criteria.

To perform a query

1. Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

2 Invoke the NewQuery method.

Example (where you want to query on a field in the SMC Opportunity View):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Opportunity+List+App
let&SWEView=SMC+Opportunity+View&SWENeedContext=false&SWEReqRowI
d=0&SWEBID=-1&SWEMethod=NewQuery&SWESetMarkup=XML
```

3 Invoke the ExecuteQuery method and specify a value to indicate the search criteria.

Example (where you want to query for a record name called IP_Webserver):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Opportunity+List+App
let&SWEView=SMC+Opportunity+View&SWENeedContext=false&SWEReqRowI
d=0&SWEBID=-
1&SWEMethod=ExecuteQuery&SWESetMarkup=XML&Name=IP_Webserver
```

For a complete list of the view and applet names to which you can navigate, see Appendix A, "Siebel Mobile Connector Application Definition Reference."

Drilling Down on Items

You can drill down on a field by specifying the name of the applet field on which you want to drill down. The detailed information about the field is retrieved from the repository. In this way you can retrieve detailed information about specific items in applets on which you have queried.

To drill down on an item

1 Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

2 Invoke the DrillDown method and pass the value of the field you want to drill down to in the SWEField argument.

Example (to drill down on the Account field):

```
http://localhost/smc_enu/start.swe?
SWECmd=InvokeMethod&SWEApplet=SMC+Opportunity+List+Applet&SWEVie
w=SMC+Opportunity+View&SWERowId=99-
27NLD&SWENeedContext=true&SWEReqRowId=1&SWEMethod=Drilldown&SWEF
ield=Account&SWESetMarkup=XML
```

NOTE: If you want to drill down into a specific record, then specify the SWERowId parameter of the row you want to drill down to. When the SWERowId parameter is not supplied in a drill down, then SWE returns the first record in the list.

For a complete list of the field names on which you can drill down, see Appendix A, "Siebel Mobile Connector Application Definition Reference."

Executing PDQs

You can execute pre-defined queries from your applications. You must invoke the ExecuteNamedQuery method and pass the name of the PDQ you want to apply.

To execute a PDQ

1. Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.
2. Invoke the ExecuteNamedQuery method and pass the value of the PDQ you want to use.

   Example (to use the My Activities for the Week PDQ):

   ```
   http://localhost/smc_enu/
   start.swe?SWECmd=ExecuteNamedQuery&SWEView=SMC+Activity+View&SWE
   NeedContext=false&SWEQueryName=My+Activities+for+the+Week&SWEBID
   =-1&SWESetMarkup=XML
   ```

Retrieving Large Data Sets

It may be necessary for an application to retrieve a very large set of data, or to obtain all the records in a set. You can retrieve large data sets by using the SWESetRowCnt parameter in your request to SWE. Set the parameter to a large number such as 100 to obtain up to 100 records. If you set the SWESetRowCnt parameter to a large number, it will take longer to get a response back from SWE and the performance may not be acceptable to your end user if this is a real-time action taken on their behalf. To improve performance, you can set the SWESetRowCnt parameter to a smaller number, but check the results to see if there are additional records. If there are additional records, you can make additional requests to SWE. If there are more rows to bring back, invoke the GoToNextSet method.

To retrieve all the records in a large set

1. Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65 and"Navigating to a Screen" on page 67.

2. Navigate within the screen to which you want to query. If there are more than 100 records, you can set the row count to 100 to get the first set of 100 records.

   Example (where you are navigating to the Contact Attachment View):

   ```
   http://localhost/smc_enu/
   start.swe?SWECmd=GotoPageTab&SWEScreen=SMC+Account+Screen&SWENee
   dContext=false&SWEBID=-1&SWESetMarkup=XML&SWESetRowCnt=100
   ```

3. Examine the XML page that is returned from SWE. Note that the APPLET tag contains the ROW_COUNTER attribute that indicates whether or not there are additional records.

   Example (where the ROW_COUNTER indicates that there are additional records by containing a " + " sign):

   ```
   <APPLET MODE="Edit" ROW_COUNTER="1 of 100+" NO_INSERT="FALSE"
   ACTIVE="FALSE" CLASS="CSSFrameBase" TITLE="Account" ID="1"
   NO_MERGE="FALSE" NO_DELETE="FALSE" NO_UPDATE="FALSE"
   NO_EXEC_QUERY="FALSE" NAME="Account Form Applet">
   ```

4 Query again, this time invoking the GoToNextSet method to obtain the next set of records.

Example:

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Account+List+Applet&
SWEView=SMC+Account+View&SWENeedContext=false&SWEReqRowId=0&SWEB
ID=-1&SWEMethod=GotoNextSet&SWESetMarkup=XML
```

5 Again, examine the XML page that is returned from SWE. Note that the APPLET tag contains the ROW_COUNTER attribute that indicates whether or not there are additional records.

Example (where the ROW_COUNTER indicates that there are additional records by containing a " + " sign):

```
<APPLET MODE="Edit" ROW_COUNTER="101 of 200+" NO_INSERT="FALSE"
ACTIVE="FALSE" CLASS="CSSFrameBase" TITLE="Account" ID="1"
NO_MERGE="FALSE" NO_DELETE="FALSE" NO_UPDATE="FALSE"
NO_EXEC_QUERY="FALSE" NAME="Account Form Applet">
```

6 To get all the records in this set, continue querying in this way until there are no additional records returned (that is, when the ROW_COUNTER attribute does not contain a " + " sign).

Parsing Dates on Records

Applications may need to parse the dates and/or times on records to perform operations on the data. Many fields contain date stamp information according to formats specified in the Siebel application definition. To parse dates on records, your application must read the format specified in the FORMAT attribute.

Examples (where date format is M/D/YYYY hh:mm:ss p and M/D/YYYY hh:mm p):

```
<CONTROL HTML_TYPE="Field" CAPTION="Created" SCALE="0" DATATYPE="utcdatetime"
HIDDEN="TRUE" NUMBER_BASED="FALSE" ID="1310" TYPE="TextBox" REQUIRED="TRUE"
TEXT_BASED="FALSE" FORMAT="M/D/YYYY hh:mm:ss p" CALCULATED="FALSE" ENABLED="TRUE"
MAX_LENGTH="32" NAME="Created">12/31/1979 04:00:00 PM</CONTROL>
```

```
<CONTROL HTML_TYPE="Field" CAPTION="Start" SCALE="0" DATATYPE="utcdatetime"
HIDDEN="FALSE" NUMBER_BASED="FALSE" ID="1801" TYPE="TextBox" REQUIRED="FALSE"
TEXT_BASED="FALSE" FORMAT="M/D/YYYY hh:mm p" CALCULATED="FALSE" ENABLED="TRUE"
MAX_LENGTH="32" NAME="Planned">8/12/1999 03:00 PM</CONTROL>
```

NOTE: The FORMAT attribute uses the standard Siebel date format specification. For example, to indicate the month of March, a single capital M indicates that the month is represented by "3"; MM indicates "03"; MMM indicates "Mar"; and MMMM indicates "March."

Retrieving Data from Hidden Fields

In Siebel application definitions, some form applets are not entirely visible by default. On the user interface, the user must click the toggle button to switch between views of the form applet. When retrieving data from these forms, by default SWE will only return data from the visible fields. If you want data from the hidden fields, use the ToggleLayout command. The following example shows a SWE request for toggling the layout.

```
http://localhost/
start.swe?SWECmd=InvokeMethod&SWEApplet=Account+Entry+Applet&SWEView=Account
Attachment+View&SWERowId=99-
28B1T&SWENeedContext=true&SWEReqRowId=0&SWEMethod=ToggleLayout&SWESetMarkup=XML
```

Updating and Synchronizing Data

This section describes how to update and synchronize data by using the XML Web Interface.

Adding Records

To add records to a list, you must first navigate to a screen that allows rows to be inserted. Then, you must send requests to SWE to execute a new record and write the data to the record. The commands used are NewRecord and WriteRecord.

To add a record

1. Login to SWE and navigate to the screen to which you want to go. For more information, see "Logging In" on page 65 and "Navigating to a Screen" on page 67.

2 Execute a new record. You must use the NewRecord command.

Example:

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Contact+List+Applet&
SWEView=SMC+Contact+View&SWENeedContext=true&SWEReqRowId=0&SWEMe
thod=NewRecord&SWESetMarkup=XML
```

NOTE: For a NewRecord command, the SWERowID parameter should be empty. The XML returned from this command will contain the new RowID for the record created. You will use the new RowID value returned from SWE in the next step. For an example of XML output from SWE that contains a RowID, see the example in Step 5 on page 80.

3 Fill in the fields in the user interface, and then write the data to the record. You must use the WriteRecord command.

Example:

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Contact+List+Applet&
SWEView=SMC+Contact+View&SWERowId=99-
4CESH&SWENeedContext=true&SWEReqRowId=1&SWEMethod=WriteRecord&La
st+Name=Haven&First+Name=Chris&SWESetMarkup=XML
```

Modifying Records

To modify a record, you must first navigate to a screen that allows records to be modified. Then, you need to perform a new query, execute the query, invoke the edit record method and write the record.

To modify a record

1 Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

2 Query for the record you want to modify. For more information, see "Querying Items" on page 68.

Caution: If you do not use a primary key to perform the query, several records may be returned in the response. There is a chance that the record you want to modify is not the one selected.

3 Write the record. You must invoke the WriteRecord method to modify the record.

Example (where you want to modify a Job Title to read "QA"):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Contact+List+Applet&
SWEView=SMC+Contact+View&SWERowId=99-
27NLD&SWENeedContext=true&SWEReqRowId=1&SWEMethod=WriteRecord&SW
ESetMarkup=XML&Job+Title=QA
```

Example (where you want to modify a record with the fields Job Title, Work Phone #, and Email Address):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Contact+List+Applet&
SWEView=SMC+Contact+View&SWERowId=99-
27NLD&SWENeedContext=true&SWEReqRowId=1&SWEMethod=WriteRecord&SW
ESetMarkup=XML&Job+Title=QA+Engineer&Work+Phone
#=4255551212&Email+Address=someone@siebel.com
```

Deleting Records

To delete a record, you must first navigate to a screen that allows records to be modified. Then, you must perform a new query, execute the query and delete the selected record.

To delete a record

1 Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

2 Query for the record you want to delete. For more information, see "Querying Items" on page 68.

Caution: If you do not use a primary key to perform the query, several records may be returned in the response. There is a chance that the record you want to delete is not the one selected.

3 Delete the selected record. You use the DeleteRecord to access the record by its primary key (in this case, the RowID).

Example:

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=SMC+Contact+List+Applet&
SWEView=SMC+Contact+View&SWERowId=0-
10LMD&SWENeedContext=true&SWEReqRowId=0&SWEMethod=DeleteRecord&S
WEC=5
```

Synchronizing Records One-Way

In some instances, a third-party application may require that a local copy of a subset of Siebel data be synchronized with data residing on the third-party application server. Such one-way synchronization of records must be handled by third-party applications. It is necessary for such applications to track the date and time that a user last synchronized data on the mobile device with the application server, and the date stamp must be passed in the GetSMCUpdate method to obtain any changed records since the date stamp.

An example of how synchronization could be used is a voice application. In order to recognize unique names or words (such as "accounts" or "contacts"), the speech recognition system must compile these words into phonemes. This is usually done on a batch process based on how frequently the data is expected to change. Using the GetSMCUpdate method, you can compile a smaller subset of information for a user giving them the ability to get real-time information from the Siebel eBusiness application.

For more information about the GetSMCUpdate method, see "GetSMCUpdate" on page 83.

NOTE: The ability to automatically synchronize data records is a feature that may be available in future releases of Siebel Mobile Connector.

Uploading Files

Applications may require that files be uploaded to the Siebel database. For example, mobile voice applications may enable users to update descriptions or add comments by capturing speech in an audio file and attaching it to the record.

NOTE: This procedure cannot be done by sending HTTP requests in a browser. Instead, uploading files must be done programmatically, so the application that uploads the files can modify the Content-Type of the HTTP request and send the file according in the appropriate format for file uploads.

To upload a file

1. Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

   The Content-Type of the HTTP request must be set to application/x-www-form-urlencoded for each step of this procedure, except where noted.

2. Create a new record. You must invoke the NewRecord method to make a record for the file you want to attach.

   Example (where Content-Type is application/x-www-form-urlencoded):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=Account+Attachment+Apple
t&SWEView=Account+Attachment+View&SWERowId=&SWENeedContext=true&
SWERowIds=SWERowId0=10-
5NIG6U&SWEReqRowId=0&SWEMethod=NewRecord&SWEC=3&SWESetMarkup=XML
```

NOTE: The SWERowId parameter is not required here and can be empty (a value for this parameter will be returned). However, SWERowId0 is a required parameter. SWERowID0 is the id of the parent row and SWERowId is the child's id. The Row IDs of the parent and child can be obtained from the XML output returned from the previous request.

3 Edit the field. You must invoke the EditField method to modify the record for the file you want to attach.

Example (where Content-Type is application/x-www-form-urlencoded):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=Account+Attachment+Apple
t&SWEW=0&SWEView=Account+Attachment+View&SWERowId=&SWEField=Accn
tFileName&SWEDIC=true&SWENeedContext=true&SWERowIds=SWERowId0=10
-
5NIG6U&SWEH=0&SWEReqRowId=1&SWESP=true&SWEMethod=EditField&SWEC=
4&SWESetMarkup=XML
```

NOTE: You will need to supply the SWERowID parameter returned in the XML output from the previous step. SWE requests that require a RowID will not work if the SWERowId parameter is not supplied.

4 Attach the file to the record. You must invoke the WriteRecord method. Set the Content-Type of the HTTP request containing the attachment to multipart/form-data for this step of the procedure.

Example (where Content-Type is multipart/form-data):

```
Content-Type: multipart/form-data; boundary=BbC04y

--BbC04y

   Content-Disposition: name="SWECmd"

InvokeMethod
```

```
--BbC04y

Content-Disposition: name="SWEApplet"

File Popup Applet

--BbC04y

Content-Disposition: name="SWEView="

Account Attachment View

--BbC04y

Content-Disposition: name="SWERowId"

10-5O167A

--BbC04y

Content-Disposition: name="SWENeedContext"

false

--BbC04y

Content-Disposition: name="SWERowIds"

SWERowId0=10-5NIG6U

--BbC04y

Content-Disposition: name="SWEReqRowId"

1

--BbC04y

Content-Disposition: name="SWEP"

19_Account+Attachment+Applet9_EditField9_10-
5O167AD_AccntFileName1_4

--BbC04y

Content-Disposition: name="SWEMethod"

WriteRecord
```

```
--BbC04y

Content-Disposition: name="SWESetMarkup"

XML

--BbC04y

Content-Disposition: name="SWEC"

4

--BbC04y

Content-Disposition: name="_SweFileName"; filename="file1.txt"

   Content-Type: text/plain

... contents of file1.txt ...

--BbC04y--
```

Tip: For additional information about HTTP file uploads, the relationship between "multipart/form-data" and other content types, performance issues, and so on, see the documentation available at http://www.w3.org/.

5 After you upload the file, examine the XML content of the user data you get back from SWE and make a note of the SWERowId parameter. You will need the value of this parameter in the next step.

Example (where the RowId is 10-5O167A):

```
- <ROW ROWID="10-5O167A" SELECTED="TRUE">

<FIELD VARIABLE="AccntFileName" NAME="AccntFileName" />

  <FIELD VARIABLE="AccntFileSize" NAME="AccntFileSize" />

  <FIELD VARIABLE="AccntFileExt" NAME="AccntFileExt" />

  <FIELD VARIABLE="AccntFileDate" NAME="AccntFileDate" />

  <FIELD VARIABLE="AccntDockStatus" NAME="AccntDockStatus"><img
width="11" alt="ENU_white.gif" src="images/white.gif" border="0"
height="11"/></FIELD>
```

```
  <FIELD VARIABLE="AccntFileDockReqFlg"
NAME="AccntFileDockReqFlg" />

  <FIELD VARIABLE="AccntFileAutoUpdFlg"
NAME="AccntFileAutoUpdFlg" />

  <FIELD VARIABLE="Comment" NAME="Comment" />

  </ROW>
```

Make a note that SWERowId = 10-5O167A. Note that in the XML output that there is no data in the FIELD tags because this is a new form and the user has not entered any data.

Tip: After you upload the file, you can also verify that the filename is present in the XML file you get back from SWE.

6 Save the record. You must invoke the WriteRecord method with the value of the SWERowId parameter that you noted in the previous step.

Example (where Content-Type is application/x-www-form-urlencoded):

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=Account+Attachment+Apple
t&SWEView=Account+Attachment+View&SWERowId=10-
5O167A&SWENeedContext=true&SWERowIds=SWERowId0=10-
5NIG6U&SWEReqRowId=1&SWEMethod=WriteRecord&SWEC=5&SWESetMarkup=X
ML
```

NOTE: Be sure to set the HTTP Content-Type to application/x-www-form-urlencoded for this step.

Downloading Files

Applications may require that files be downloaded from the Siebel database to a mobile device or other platform. For example, mobile voice applications may enable users to listen to voice recordings stored as audio files.

To download a file

1 Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

   The Content-Type of the HTTP request must be set to application/x-www-form-urlencoded for each step of this procedure, except where noted.

2 Drill down in the record containing the attached file. You must invoke the Drilldown method and pass the value of the SWERowId parameter.

   Example:

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEApplet=Account+Attachment+Apple
t&SWEView=Account+Attachment+View&SWERowId=10-
5O167A&SWENeedContext=true&SWERowIds=SWERowId0=10-
5NIG6U,SWERowId1=&SWEReqRowId=1&SWEMethod=Drilldown&SWEC=3&SWEFi
eld=AccntFileName&SWESetMarkup=XML
```

NOTE: SWERowID0 is the Row ID of the parent row and SWERowId is the child's ID. The Row IDs of the parent and child can be obtained from the XML output returned from the previous request. SWE requests that require a Row ID will not work if the SWERowId parameter is not supplied.

Getting Updates

In some instances, a third-party application may require a local copy of a subset of Siebel data for its own use. Such applications can periodically check if there is any new data, and retrieve just the data that has changed. The GetSMCUpdate method is used to obtain such updates. This method extracts the system level record information that is otherwise not available through the user interface.

NOTE: Generally, when using Siebel Mobile Connector you can set the SWESetMarkup parameter to HTML, WML, or XML. However, you may not use the GetSMCUpdate method or receive alerts in any other markup than XML.

GetSMCUpdate

The method GetSMCUpdate is invoked to retrieve the changes in data from a specified time to the current time. The changes that are noted are at the record level versus for a specific field. Even if you apply a style sheet which parses the particular field that changes, the record will show as having changed. If your application requires notification about changes to specific fields, then you can use the Alert Business Service for this purpose. This is an important point for understanding this method.

The GetSMCUpdate method cannot be subscribed to at the Form Applet or Entry Applet level, only at the List Applet level. Also, if a field that has changed is present in both the List Applet and the Form Applet, then the GetSMCUpdate method will pick-up the changes at the record level in the Form Applet.

When using the GetSMCUpdate method to retrieve updates for child applets, the method only returns the records for the child applet associated with a particular parent record.

NOTE: For a client application to get updates, it is also necessary that it has set the parameter VoiceApplication = TRUE. For more information, see "Configuring a User Agent for Siebel Mobile Connector Applications" on page 37.

To get an update, send a command to SWE with the following parameters:

Parameters	Required/ Optional	Description
SWECmd = InvokeMethod	Required	Sets the name of the SWE command to InvokeMethod.
SWEMethod = GetSMCUpdate	Required	Sets the name of the SWE method to GetSMCUpdate. This is a new SWE method available with the Siebel Mobile Connector.
SWEView = < view_name >	Required	Specifies the name of the view you want to receive an update for.
SWEApplet = < applet_name >	Required	Specifies the name of the applet you want to receive an update for. You must subscribe to the GetSMCUpdate at the List Applet level, not the Form Applet or Entry Applet level.
SWESetMarkup = XML	Required	Specifies the XML markup mode.
LastUpdate = < date >	Optional	Specifies the last update date or date/time. For example, 07/30/2002 or 07/30/2002 12:00:00. The time is denoted in 24 hour format (military time). The update is the delta between the current time and the time specified. If no value is specified, all requested data will be returned.
SWEXslStyleSheet = < name_of_style_sheet >	Optional	Specifies an XSL style sheet to be applied to the output results.
SWESetRowCnt = < #_of_rows >	Optional	Specifies the number of rows to be applied. If no value is given, the number of rows returned matches the default value for NumberOfListRows specified in the smc.cfg file.

To get an update

1 Login to SWE and navigate to the screen, view and applet to which you want to go. For more information, see "Logging In" on page 65, "Navigating to a Screen" on page 67 and "Navigating within a Screen" on page 68.

NOTE: It is not necessary to navigate to a specific view or applet if you want to get an update for the screen's default view.

2 Invoke the GetSMCUpdate method.

Example (where you want to get an update for records that have changed from 07/01/2002 to the present):

```
http://localhost/sales/start.swe?
SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+Account+V
iew&SWEApplet=SMC+Account+ListApplet&LastUpdate=07/01/
2002&SWESetRowCnt=100&SWESetMarkup=XML
```

NOTE: If you pass an invalid date for the LastUpdate parameter, you may receive an error message "Unable to load message 0xffff." This message indicates that date given is not a valid parameter for the GetSMCUpdate method.

Additional Examples

Here are several additional example requests to SWE using the GetSMCUpdate method.

The following request passes a date as the value of the LastUpdate parameter. It retrieves all records that have changed since 07/30/2002 or later.

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+A
ccount+View&SWEApplet=SMC+Account+List+Applet&SWESetMarkup=XML&Las
tUpdate=07/30/2002
```

The following request passes a date and time as the value of the LastUpdate parameter. It retrieves all records that have changed since 07/30/2002 at noon or later.

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+A
ccount+View&SWEApplet=SMC+Account+List+Applet&SWESetMarkup=XML&Las
tUpdate=07/30/2002 12:00:00
```

The following request passes a date as the value of the LastUpdate parameter and uses a style sheet to filter data. It retrieves only the data specified in CompanyName_SiebelMobileConnector_SMCAccountView_GM.xsl for records that have changed since 07/30/2002 or later.

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+A
ccount+View&SWEApplet=SMC+Account+List+Applet&SWESetMarkup=XML&Las
tUpdate=07/30/2002
12:00:00&SWEXslStyleSheet=CompanyName_SiebelMobileConnector_SMCAcc
ountView_GM.xsl
```

NOTE: Remember that the GetSMCUpdate method retrieves changes at the record level and not for specific fields. If you apply a style sheet to limit data to a specific field, this will not make a difference. When your application must retrieve updates at the field level not the record level, then use the Alert Business Service.

The following request passes a date as the value of the LastUpdate parameter. It retrieves all records that have changed since 07/30/2002 or later, up to a maximum number of 50 records.

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+A
ccount+View&SWEApplet=SMC+Account+List+Applet&SWESetMarkup=XML&Las
tUpdate=07/30/2002&SWESetRowCnt=50
```

The following request passes a date as the value of the LastUpdate parameter. It retrieves all records that have changed since 07/30/2002 or later (assuming that there are less than 500 records).

```
http://localhost/smc_enu/
start.swe?SWECmd=InvokeMethod&SWEMethod=GetSMCUpdate&SWEView=SMC+A
ccount+View&SWEApplet=SMC+Account+List+Applet&SWESetMarkup=XML&SWE
SetRowCnt=500
```

Global Language Support

Siebel applications are designed to meet the needs of customers operating in a global environment. The Siebel Mobile Connector API has an English interface that can be used to create applications in multiple languages to meet the needs of customers.

The management of different languages in third-party applications is a task that must be handled by the middleware application server and its communication to various devices.

For general information about deploying Siebel applications in a global environment, see *Global Deployment Guide* in the *Siebel Bookshelf*.