Siebel CTI Administration Guide

Siebel Innovation Pack 2013

Version 8.1/8.2

September 2013

ORACLE®

Copyright © 2005, 2013 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

1 What's New in This Release

Siebel CTI Administration Guide describes how to configure Siebel CTI (computer telephony integration) for Siebel Business Applications.

What's New in Siebel CTI Administration Guide, Version 8.1/8.2

Table 1 lists the changes described in this version of the documentation to support this release of the software. The new features described in Table 1 are available in Siebel CRM version 8.1.1.11, Siebel CRM version 8.2.2.4, and later.

Table 1. What's New in Siebel CTI Administration Guide, Version 8.1/8.2

Topic	Description
"Parameters for Communications Configurations" on page 47	Modified topic: ■ Removed the description of the parameter Accessibility. This parameter no longer applies. Accessibility for the communications toolbar is always enabled for deployments using Siebel Open UI. ■ Added a description of the new parameter RetryBeforeStopPush. ■ Added a description of the new parameter StopPushString.

What’s New in Siebel CTI Administration Guide, Version 8.1, Rev. B

Table 2 lists the changes described in this version of the documentation. (Version 8.1, Rev. B of this guide was for Siebel CRM version 8.1.1.10.)

Table 2. What’s New in Siebel CTI Administration Guide, Version 8.1, Rev. B

Topic	Description
“Parameters for Communications Configurations” on page 47	Modified topic: ■ Added a description of the new parameter Accessibility. This parameter is available only for deployments using Siebel Open UI. For Siebel Open UI, accessibility is enabled for the communications toolbar by default. (This parameter no longer applies, as of version 8.1.1.11 and later. Accessibility for the communications toolbar is always enabled.) ■ Added a description of the new parameter AutoLoadDriver. The ability to set this parameter to False is available only for deployments using Siebel Open UI. ■ Updated the description of the parameter CallFromUICommand. Some new functionality applies only to deployments using Siebel Open UI.
“Preference Settings for Communications” on page 218	Modified topic. Updated the information for the Sound File user preference.

What’s New in Siebel CTI Administration Guide, Version 8.2, Rev. B

Table 3 lists the changes described in this version of the documentation. (Version 8.2, Rev. B of this guide was for Siebel CRM version 8.2.2.3.)

Table 3. What’s New in Siebel CTI Administration Guide, Version 8.2, Rev. B

Topic	Description
“Parameters for Communications Configurations” on page 47	Modified topic: ■ Added a description of the new parameter Accessibility. This parameter is available only for deployments using Siebel Open UI. For Siebel Open UI, accessibility is enabled for the communications toolbar by default. (This parameter no longer applies, as of version 8.2.2.4 and later. Accessibility for the communications toolbar is always enabled.) ■ Added a description of the new parameter AutoLoadDriver. The ability to set this parameter to False is available only for deployments using Siebel Open UI. ■ Updated the description of the parameter CallFromUICommand. Some new functionality applies only to deployments using Siebel Open UI.
“Preference Settings for Communications” on page 218	Modified topic. Updated the information for the Sound File user preference.

Additional Information

User interface references in this guide assume that you are using Siebel Open UI.

NOTE: The procedures in this guide assume that you do not use left-hand navigation. However, you can set up left-hand navigation. For more information about left-hand navigation and about implementing it, see *Siebel Fundamentals for Siebel Open UI*.

2 Overview of Siebel CTI and Related Products

This chapter provides an overview of Siebel CTI and related products. It includes the following topics:

About Siebel CTI and Siebel Communications Server

Siebel Business Applications provide an infrastructure to support communications functionality for Siebel application users, including *Siebel CTI* (Computer Telephony Integration). Siebel CTI works with CTI middleware products, which integrate with telephony systems so that agents can make or receive voice calls through the Siebel application.

Other types of communications can also be supported for some deployments. For more information about products that use Siebel Communications Server, see "Siebel Product Modules That Use Siebel Communications Server" on page 16.

The term *Siebel Communications Server* refers to the set of Siebel Server components and related features that support the functionality described in this book and the functionality described in *Siebel Chat Guide* and *Siebel Email Administration Guide*. The key server components are part of the Communications Management component group. Communications drivers provide the interface to the communications system with which you are integrating. Several drivers are provided with Siebel Business Applications. Where appropriate, you can create custom drivers for third-party products, by using the Adaptive Communications API.

The types of communications supported by Siebel Communications Server include the following:

- **Session-based or interactive communications.** Supports multichannel interactive communications for call or contact center agents who use the communications toolbar to make or receive voice calls through Siebel CTI and CTI middleware products. This capability is described in this book. Siebel Chat, which is described in *Siebel Chat Guide*, also uses session-based communications.
- **Inbound communications.** Supports integrating to third-party email servers and processing inbound email, for Siebel Email Response deployments. For more information, see *Siebel Email Administration Guide*.
- **Outbound communications.** Supports integrating to a variety of third-party communications systems such as email servers to send outbound communications. For more information, see *Siebel Email Administration Guide*.

NOTE: Your Siebel implementation might not have all the features described in this guide, depending on which software modules you have purchased.

Siebel Product Modules That Use Siebel Communications Server

This topic provides high-level information about how Siebel Communications Server works with closely related product modules. Several Siebel modules use the Communications Server infrastructure to support certain functionality for the Siebel Business Applications.

Communications Server functionality is not limited to what is described in the following topics. Many specific features of Siebel Business Applications that involve Siebel Communications Server are discussed throughout this book. However, some of these features are described in greater detail in *Siebel Email Administration Guide*.

For example, customers using an application like Siebel Call Center might implement Siebel CTI or Siebel Email Response. You can use the customer dashboard feature and Siebel SmartScript with these modules. You might also implement Siebel Communications Server in order to support the Send Email or Send Fax commands.

This topic contains the following information:

- "About Siebel CTI" on page 16
- "About Siebel Chat" on page 17
- "About Siebel Email Response and Outbound Communications" on page 17
- "About Siebel Workflow" on page 17

About Siebel CTI

Siebel CTI (Computer Telephony Integration) provides voice-channel support for call center agents using Siebel Business Applications. Integration with a third-party CTI middleware package provides CTI capabilities. Use communications administration views to configure aspects of the integration, such as to define telesets and associate them with agents.

The Communications Session Manager handles CTI communications. Agents are notified of incoming calls through the communications toolbar and can perform a range of call-handling activities using the toolbar and related menu commands.

Call routing can be handled by the ACD (Automatic Call Distributor), by the CTI middleware, or by a third-party vendor's queuing engine that has been integrated with Siebel Communications Server.

CTI functionality is optionally available through Oracle Contact On Demand (serving as the CTI middleware) and the Virtual CTI communications driver. For more information about this driver, see Chapter 10, "Using the Virtual CTI Driver."

Where appropriate, you can create custom drivers for third-party CTI middleware products, by using the Adaptive Communications API. For more information, see Appendix A, "Developing a Communications Driver."

About Siebel Chat

Siebel Chat is a customer-contact channel whereby users in real time can chat with a customer service representative (otherwise known as an agent). Siebel Chat can use Oracle Contact On Demand and the Virtual CTI communications driver or use a custom communications driver with a third-party chat product. For more information about this driver, see Chapter 10, “Using the Virtual CTI Driver.” For more information about Siebel Chat, see *Siebel Chat Guide*.

About Siebel Email Response and Outbound Communications

Siebel Email Response allows agents to receive and reply to inbound email messages. Siebel Business Applications provide communications drivers that integrate with a variety of email servers that support SMTP, IMAP, and POP3 protocols. Communications administrators configure profiles, response groups, and workflows to determine how to process inbound email messages. You can configure Siebel Assignment Manager, with Siebel Workflow, to route inbound email.

The Communications Inbound Receiver and Communications Inbound Processor components receive and process inbound email communications. The Communications Outbound Manager component sends outbound email for agent replies and other outbound communications. To enable inbound and outbound communications functionality, you must configure profiles for one of the communications drivers for email. For more information about Siebel Email Response and other outbound communications capabilities, see *Siebel Email Administration Guide*.

Outbound communications capabilities include the following:

- Agents can send email replies to inbound messages, using Siebel Email Response.
- Agents and other Siebel application users can use the Send Email, Send Fax, and Send Wireless Message commands. Alternatively, you can configure Send Email as a client integration that does not use the Communications Outbound Manager server component. (Send Page is also available but uses the Page Manager component.)
- The ability to send communications content to designated recipients using outbound communication requests. Users can create and submit communication requests manually through the user interface, or requests can be created and submitted programmatically. Several Siebel modules invoke business service methods through workflows to send outbound communications.

About Siebel Workflow

Siebel Workflow is a Siebel CRM module that can work closely with Siebel Communications Server. Business service methods, such as those for the Communications Client business service, can be invoked from workflow processes to serve a variety of purposes.

For more information, see "Using Business Services with Siebel Communications Server" on page 181 and Appendix B, "Siebel Communications Server Business Services." For more information about workflow processes and design, see *Siebel Business Process Framework: Workflow Guide*. Outbound communications capabilities can also be accessed through business services and workflow processes. For more information about workflow processes and Siebel Email Response, see *Siebel Email Administration Guide*.

About Siebel Communications Server Architecture

Figure 1 illustrates the overall architecture of the communications infrastructure for the Siebel Business Applications, in particular for Siebel CTI deployments. For information about how email, fax, and paging are integrated into the communications infrastructure, see *Siebel Email Administration Guide*.

NOTE: The Siebel Developer Web Client is supported for administration, development, and troubleshooting usage scenarios only.

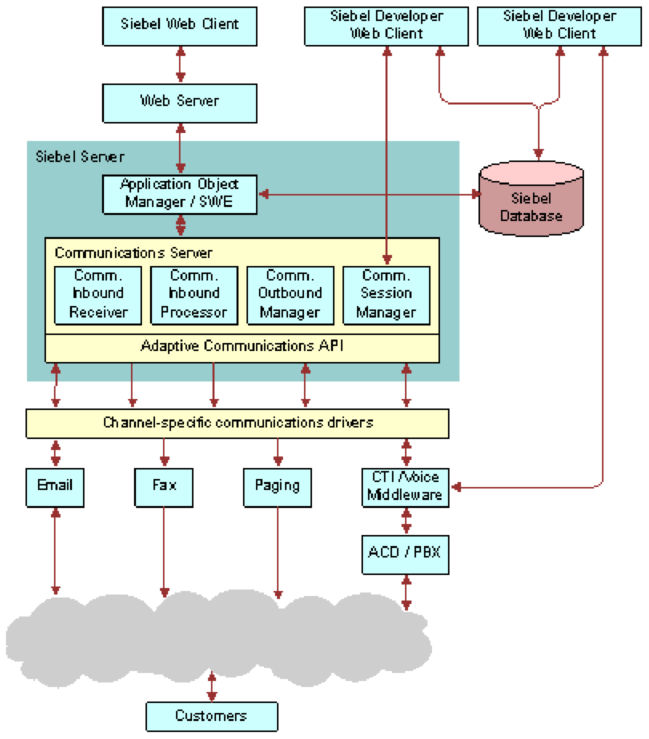

Figure 1. Architecture of Communications Infrastructure

About Communications Configuration Data

The Administration - Communications screen allows administrators to configure several kinds of communications data.

Sample communications configuration data is provided in the Siebel database. You can modify the preconfigured (out of the box) data, or create new data, to customize Siebel Communications Server and the Siebel Business Applications to support the communications channels you use and to support your business processes. You can also find the sample communications configuration data in the Siebel Sample Database.

This topic contains the following information:

- "Types of Communications Configuration Data" on page 19
- "Communications Configuration Requirements for Siebel Modules" on page 22

Types of Communications Configuration Data

Figure 2 shows key kinds of configuration data that you work with in the Administration - Communications screen. This data enables you to specify communications functionality within the Siebel Business Applications. One-to-one, one-to-many, and many-to-many relationships between elements are represented by the connector lines.

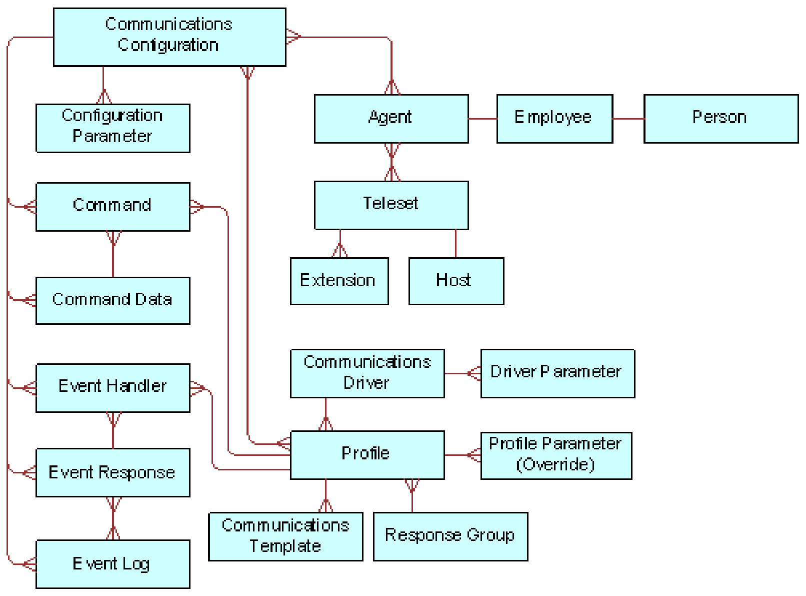

Figure 2. Overview of Communications Configuration Elements

Table 4 on page 20 identifies communications elements that you create in the process of configuring Siebel Communications Server, indicating where the element is created, configured, or specified. Unless otherwise noted, the application views identified are in the Administration - Communications screen. For more information about communications templates and response groups, which are shown in Figure 2 on page 19, see *Siebel Email Administration Guide*.

This topic is part of "About Communications Configuration Data" on page 19.

Table 4 describes the major Siebel CTI configuration elements in relation to other elements and identifies the Siebel application view in which each type of element is created or configured. See also Figure 2 on page 19.

Table 4. Communications Configuration Elements

Configuration Element	Description	How Related to Other Configuration Elements	Siebel View for Creating or Configuring Element
Communications configuration	A set of related data for communications, used for agents working with channels such as voice or email. Includes parameters.	Parent of event handler, event response, event log, command, and command data. Associated with profile and agent.	All Configurations
Communications driver (Drivers are database records that reference files such as library files.)	Usually a library file that Communications Server uses to interface to a communications system, such as voice or email. Includes parameters.	Parent of profile.	Communications Drivers and Profiles
Profile	A definition that provides access to a communications driver for specific contexts. Includes overrides for driver parameter.	Child of communications driver. Associated with configuration, event handler, and command.	Communications Drivers and Profiles
Agent	A contact center agent, manager, or other user in a communications configuration. Can optionally use multiple communications channels.	Derived from employee. Associated with configuration, teleset (for voice agents), and ACD queue (for voice agents).	All Configurations Agent General Profile All Telesets

Table 4. Communications Configuration Elements

Configuration Element	Description	How Related to Other Configuration Elements	Siebel View for Creating or Configuring Element
ACD queue (agent queue)	A queue on the ACD that agents can log in to. Agents can also log in to multiple queues. Agents can log in to queues automatically, from the communications toolbar, or from the User Preferences screen.	Associated with agent (for voice agents).	Administrators: ■ List of Values (Administration - Data screen) ■ Agent General Profile (Administration - Communications screen) End users (User Preferences screen): ■ Communications
Teleset	A physical phone device such as one used in a call center. Optionally, you can specify a host name for shared telesets used in hoteling scenarios.	Parent of extension. Associated with agent (for agents using the voice channel).	All Telesets
Extension	An extension for a teleset.	Child of teleset.	All Telesets
Event handler	A definition that provides a means of evaluating information received from the communications driver. Includes parameters.	Child of configuration. Associated with profile and event response.	All Configurations All Event Handlers
Event response	A definition of the response invoked by the event handler. Includes parameters.	Child of configuration. Associated with event handler and event log.	All Event Responses
Event log	A definition that specifies logging behavior for event response. Includes parameters.	Child of configuration. Associated with event response.	All Event Logs

Table 4. Communications Configuration Elements

Configuration Element	Description	How Related to Other Configuration Elements	Siebel View for Creating or Configuring Element
Command	A definition that provides a means of performing a communications action. Commands are typically sent to the communications driver. Includes parameters. A command can be specified as a subcommand of another command.	Child of configuration. Associated with profile and command data.	All Configurations All Commands
Command data	A definition that provides data applicable to command. Includes parameters.	Child of configuration. Associated with command.	All Command Data

Communications Configuration Requirements for Siebel Modules

Much of the communications configuration data that this chapter is concerned with, particularly for Siebel CTI, involves sets of data organized under a type of record called a *communications configuration*. Used as a noun, this term (sometimes shortened to *configuration*) refers to these records and to associated elements such as configuration parameters, agents, telesets, profiles, commands, and events.

For a Siebel CTI deployment, then you must create or use a communications configuration, such as a predefined multichannel configuration. For more information, see "About Siebel CTI" on page 16.

This topic is part of "About Communications Configuration Data" on page 19.

For information about additional kinds of configuration data related to communications, see "Types of Communications Configuration Data" on page 19.

Communications drivers and *profiles*, through which Siebel Communications Server can interface with external systems, are part of the communications configuration data. Examples of external systems include Oracle Contact On Demand, CTI middleware, and email servers. Communications drivers are usually based on library files, such as DLL (Dynamic Link Library) files on Microsoft Windows. Corresponding configuration data in the Siebel database, including driver parameters, is defined to work with each driver. Communications profiles customize driver behavior for particular purposes by allowing driver parameter values to be overridden wherever the profile is in use. Communications configurations have associated profiles.

Every deployment of Siebel Communications Server uses communications drivers and profiles. However, as shown later in this topic, some Siebel modules use communications drivers and profiles but do *not* use (or might not use) communications configurations. For more information about:

- Configuring communications drivers and profiles, see Chapter 3, "Configuring Communications Drivers and Profiles"
- Defining communications configurations and related elements such as events and commands, see Chapter 4, "Configuring Siebel CTI," and Chapter 5, "Configuring Events and Commands"
- Configuring user interface elements, see the following:
  - Chapter 6, "Configuring User Interface Elements"
  - Chapter 7, "Configuring Advanced Communications Features"

Related Books

Siebel Email Administration Guide

Configuring Siebel Business Applications

Configuring Siebel Open UI

Using Siebel Tools

Siebel Developer's Reference

Modules That Might Use Communications Configurations

Siebel Email Response customers can also use communications configurations. Note that:

- If inbound email messages are routed using an integrated package for queuing and routing, then you *must* use communications configurations.
- If inbound email messages are routed manually or using Siebel Assignment Manager, then you *cannot* use communications configurations.

Using communications configurations enables your agents to use features such as the communications toolbar and communications commands in the application-level menus. It also enables you to use an integrated queuing package to route inbound work items and to support multichannel communications capabilities.

Communications Configurations and the Send Email, Send Fax, Send Wireless Message, and Send Page Commands

The Send Email, Send Fax, Send Wireless Message, and Send Page commands are available through the communications toolbar for agents who are enabled to use interactive communications session functionality with Siebel CTI or with Siebel Email Response when used with an integrated package for queuing and routing.

For these commands to be available through the communications toolbar, employees must be specified as agents within a communications configuration, and session communications must be enabled for the Siebel application. For more information, see "Enabling Session Communications" on page 201.

By default, these commands are also available in the File application-level menu for Siebel Business Applications users. A communications configuration is not needed in order for employees to access these menu commands. Apart from Send Page, these commands use communications drivers and profiles and are available when the communications infrastructure is in place. When the Send Email command is integrated with Microsoft Outlook or Lotus Notes, it does not use communications drivers and profiles.

For more information about the Send Email, Send Fax, Send Wireless Message, and Send Page commands, see *Siebel Email Administration Guide*.

Process of Configuring Siebel CTI

To set up Siebel Communications Server, review the process described in this topic. For some of the steps in the configuration process described here, detailed instructions are provided in subsequent chapters. In general, you perform the following tasks:

1. "Determining Your Communications Deployment Needs" on page 24.
2. "Setting Up Your External Communications System" on page 25.
3. "Setting Up Your Siebel Server Software for Siebel CTI" on page 26.
4. "Defining Siebel Communications Server Configuration Data" on page 27.
5. "Putting Siebel CTI into Production" on page 29.

Related Topics

"About Communications Configuration Data" on page 19

Chapter 3, "Configuring Communications Drivers and Profiles"

Chapter 4, "Configuring Siebel CTI"

Determining Your Communications Deployment Needs

The multichannel support offered by Siebel Communications Server and its architecture requires that you plan your communications deployments carefully and thoroughly. Read all the relevant sections of this book before you start the configuration process. For more information about integrating with email, fax, and other outbound communications channels, see *Siebel Email Administration Guide*.

NOTE: If you use an external communications system, such as a CTI middleware server or email server, that is not supported by an existing communications driver, then you might have to create your own driver. For more information, see Appendix A, "Developing a Communications Driver."

This task is a step in "Process of Configuring Siebel CTI" on page 24.

Consider the following issues as you plan your implementation of Siebel Communications Server in support of Siebel Business Applications:

- Determine the Siebel applications and other products to use in your implementation. Determine the communications channels and external communications systems to use with your Siebel applications. Channels might include voice, email, fax, and so on.

  Siebel applications or product modules might include Siebel Call Center, Siebel Service, Siebel Sales, Siebel CTI, Oracle Contact On Demand, Siebel Email Response, Siebel Chat, or other applications or products. (Not all modules are available for all applications or parent modules.)

- Determine which Siebel application features, technologies, or advanced configurations to implement, such as global or distributed deployments, application integration, work-item routing, hoteling, multitenancy, the customer dashboard, and so on.

- Determine whether the third-party communications products you expect to use are supported by Oracle. For information about supported products, see My Oracle Support Certifications.

- For communications activities that make use of communications configurations, determine the number of agents for each supported channel. For Siebel CTI, Siebel Email Response, and other modules, you define agents within communications configurations.

- Determine data and application visibility issues for your users, because they relate to communications activities.

- Determine available resources in your communications systems. For Siebel CTI, determine the available telesets and extensions, hoteling workstations, ACD queues, and so on.

- Determine the work model for inbound or outbound communications in your call center or contact center: that is, the logic of each customer interaction.

- Determine your requirements for performance, scalability, and architecture, and determine your deployment plans.

  Determine the requirements in the preceding areas, because they relate to Siebel Communications Server, other Siebel modules or server components, and the Siebel client types you use. Also consider architectural issues for external communications systems such as CTI middleware and email servers. Information you gather in this process helps determine how you perform later configuration steps. For more information about planning your system architecture, see *Siebel Deployment Planning Guide* and *Siebel Performance Tuning Guide*. See also the *Siebel Installation Guide* for the operating system you are using.

- Before you begin the process described in "Defining Siebel Communications Server Configuration Data" on page 27, review how communications configurations function, and assess your requirements in this area.

Setting Up Your External Communications System

Before you can configure Siebel Communications Server for CTI, you must set up your PBX or ACD (Automatic Call Distribution) switch and install and configure the CTI middleware. For more information, see your vendor documentation. Once you have performed this preparatory task, you are ready to perform Siebel Server setup tasks.

This task is a step in "Process of Configuring Siebel CTI" on page 24.

Setting Up Your Siebel Server Software for Siebel CTI

Deploying Siebel Communications Server for Siebel CTI requires that you enable and run the following Siebel Server components:

- The Application Object Manager component for your Siebel application, which supports user sessions in the Siebel Web Client. For example, you might be running instances of the Application Object Manager for Siebel Call Center for the languages you require.
- Server Request Broker and Server Request Processor, which coordinate communications activities between instances of the Application Object Manager and instances of the Siebel Communications Server components. Server Request Broker and Server Request Processor are automatically enabled for each Siebel Server.
- Siebel Communications Server components (in the Communications Management component group):
  - Communications Session Manager (alias CommSessionMgr)
  - Communications Configuration Manager (alias CommConfigMgr) (optional)

Some minimal configuration of the preceding components might also be necessary. For more information, see Chapter 8, "Administering Siebel Communications Server." See also *Siebel System Administration Guide*. For more information about the architecture of Siebel Communications Server, see "About Siebel Communications Server Architecture" on page 18. For information about the Siebel Communications Server components that are used by Siebel Email Response and by other outbound communications features, see *Siebel Email Administration Guide*.

This task is a step in "Process of Configuring Siebel CTI" on page 24.

To set up your Siebel Server software for Siebel CTI

1. Install your Siebel Server software, as described in the *Siebel Installation Guide* for the operating system you are using. See also *Siebel System Administration Guide*.

   After installing the Siebel Server, make sure that you enable the Communications Management component group when you configure the Siebel Server.

2. Enable session communications on the Application Object Manager component.

   Do this on each Application Object Manager that is to support users for whom the communications toolbar and other interactive session communications functionality must be available. For more information, see "Enabling Session Communications" on page 201.

3. Configure each Siebel Communications Server component (in the Communications Management component group) that you require for your implementation.

   For more information, see Chapter 8, "Administering Siebel Communications Server."

Defining Siebel Communications Server Configuration Data

This topic describes in general terms how to define or modify Siebel Communications Server configuration data. Do this when all modules have been installed and the server components have been configured. You can perform many of the steps in a different sequence than that shown here. For more information about communications configurations, see "About Communications Configuration Data" on page 19. For detailed procedures for working with communications configuration data, see the applicable topics in Chapter 4, "Configuring Siebel CTI."

NOTE: When you have finished specifying the communications configuration data, you must exit and restart the Siebel application for the settings to take effect. After Siebel Communications Server has been deployed, any end users who are currently connected must also exit and restart the application for the new or modified configuration data to take effect.

This task is a step in "Process of Configuring Siebel CTI" on page 24.

To define Siebel Communications Server configuration data

1. Start the Siebel application, such as Siebel Call Center, logging in as the system administrator (for example, as the user SADMIN).
2. Navigate to the Administration - Communications screen.

   For information about the views in this screen, where you perform most of the tasks identified here, see Appendix C, "Views for Communications Administration."
3. Configure communications drivers and profiles. As applicable, associate profiles with communications configurations you have already created:
   a. For each communications driver you are using, specify default parameter values.
   b. For each communications driver you are using, create profiles. As appropriate, specify parameter values to override the driver parameters.
   c. If you are not using communications configurations, then skip to Step 11 on page 29.

      For more information about configuring drivers and profiles, see Chapter 3, "Configuring Communications Drivers and Profiles." For more information about modules that use communications configurations, see "Communications Configuration Requirements for Siebel Modules" on page 22.
4. To support Siebel CTI, create a communications configuration.

   You can modify an existing communications configuration, or create a new configuration and transfer data into it from an existing configuration. For more information about creating communications configurations, see "Creating or Modifying a Communications Configuration" on page 44.
5. For the current configuration, specify configuration parameter values.

   For more information, see "Specifying Parameters for a Communications Configuration" on page 46.
6. For the current configuration, associate profiles you created for your communications driver.

   For more information, see "Creating or Modifying a Communications Configuration" on page 44.

7. For the current communications configuration, specify the agents for whom the communications configuration apply:

   a. Add each agent to one or more configurations. If you add an agent to more than one configuration, then specify which configuration is primary for the agent.

   b. Configure the agents, such as (for voice agents) to specify which ACD queues they are to use, to define ACD agent login and password, and so on.

      NOTE: If you are not defining ACD queues, then the previous step is not necessary.

   c. If telesets have already been defined (as in Step 8), then you can specify the telesets an agent can use.

      You do not have to associate agents with telesets that are used for hoteling, or associate agents with telesets if they do not use the voice channel. For more information about specifying agents, see "Specifying Agents" on page 55.

8. For Siebel CTI (the voice channel), then specify the telesets for your call center:

   a. If a teleset is used for hoteling, then specify the host name for the hoteling computer.

   b. For each teleset, specify the extensions for the teleset.

   c. For each teleset, you can specify agents that can use the teleset.

      In order to associate an agent with a teleset, you must have already added the agent to the configuration (as in Step 7). You do not have to associate agents with telesets that are used for hoteling. For more information about telesets, see "Specifying Telesets" on page 58. See also "Configuring Telesets for Hoteling" on page 168.

9. Specify or verify event handlers and associated event responses and event logs, and associate them with the communications configuration:

   a. Create the event logs that you specify in your event responses, and associate them with the configuration.

   b. Create the event responses you need, optionally specifying one or more event logs for each event response, and associate the event responses with the configuration.

   c. Create the event handlers you need, specifying an event response for each event handler, and associate the event handlers with the configuration.

      For more information about events, see "Defining Communications Events" on page 62.

10. Specify or verify commands and command data definitions, and associate them with the communications configuration:

    a. Create the command data definitions that you specify in your commands, and associate them with the configuration.

    b. Create the commands that you need, specifying a command data definition for each command, and associate the commands with the configuration.

       For more information about commands, see "Defining Communications Commands" on page 67.

11 Complete any other needed configuration, customization, or performance-tuning tasks for your Siebel implementation, and test the communications configurations you are using. For example, you might modify the *Online Help* for your users.

After you have verified the configuration, you can put Siebel Communications Server into production.

Putting Siebel CTI into Production

This topic describes how to transfer your communications settings from the test environment to the production environment.

This task is a step in "Process of Configuring Siebel CTI" on page 24.

To put Siebel CTI into production

1. Follow the instructions for transferring your Siebel application from the test environment to the production environment.

   For more information, see *Using Siebel Tools*.

2. Configure all production instances of Siebel Server components for Communications Server, as noted in "Setting Up Your Siebel Server Software for Siebel CTI" on page 26.

3. Verify your communications configuration data in the production environment.

   For example, you might have to modify any communications configuration parameter values appropriately for a production environment.

4. Provide your end users with instructions for the following activities, as appropriate for your implementation and for the particular types of users:

   - Starting the Siebel client. For example, this can include providing the users with the URL for a Siebel Call Center application that is configured to provide access to CTI or other communications functionality.
   - Accessing the online help for the application they are using.
   - Setting communications preferences in the User Preferences screen.
   - Using user interface controls such as the communications toolbar, Communications submenu commands, or the Communications screen to perform communications-related tasks.

   For information about these tasks, see Chapter 9, "Communications Operations for End Users."

   See also documentation for related Siebel products such as Siebel Email Response.

3 Configuring Communications Drivers and Profiles

This chapter provides an overview of communications drivers and profiles used with Siebel Communications Server and describes how to configure drivers and profiles. It includes the following topics:

About Communications Drivers and Profiles

This topic provides background information about communications drivers and profiles and the roles they play in supporting Siebel Communications Server functionality. Communications drivers interface with communications software such as CTI middleware or email servers.

For more information about configuring drivers and profiles for use with email, see *Siebel Email Administration Guide*.

This topic contains the following information:

For more information:

- If you are specifying communications configurations (such as for Siebel CTI or other session communications deployments), then see Chapter 4, "Configuring Siebel CTI," and related chapters.
- For more information about configuring drivers and profiles for use with email, see *Siebel Email Administration Guide*.
- For information about configuring drivers and profiles for Siebel Chat, see *Siebel Chat Guide*.

- If you require integrating with communications systems that are not supported by drivers, then see Appendix A, “Developing a Communications Driver.”

Communications Drivers Provided with Siebel Business Applications

The following are some of the Siebel communications drivers provided with Siebel Business Applications:

- **Virtual CTI.** This driver enables CTI integration with Oracle Contact On Demand for the voice channel. This driver can also be used with Siebel Chat, which is described in *Siebel Chat Guide*. For more information, see “About Siebel CTI” on page 16 and Chapter 10, “Using the Virtual CTI Driver.”
- **Push Keep Alive.** Provides a heartbeat function to help maintain push communications for session-based communications drivers loaded by Communications Session Manager (such as the Virtual CTI driver). For more information, see “Using the Push Keep Alive Driver for Session Connections” on page 178.
- **Internet SMTP/IMAP Server.** Supports Internet mail servers that support the SMTP protocol for outbound email, fax, or wireless messages or the IMAP protocol for inbound email (for Siebel Email Response). This driver supports both plain-text and HTML email. For more information, see “About Siebel Email Response and Outbound Communications” on page 17 and see *Siebel Email Administration Guide*.
- **Internet SMTP/POP3 Server.** Supports Internet mail servers that support the SMTP protocol for outbound email, fax, or wireless messages or the POP3 protocol for inbound email (for Siebel Email Response). This driver supports both plain-text and HTML email. For more information, see “About Siebel Email Response and Outbound Communications” on page 17 and see *Siebel Email Administration Guide*.
- **Modem-based TAP Paging.** For Siebel Paging, supports paging over a modem using the Telocator Alphanumeric Paging (TAP) protocol, for outbound communication requests. Siebel Paging is included with all base applications of the Siebel Business Applications. (The Send Page command does not use this driver.) For more information, see *Siebel Email Administration Guide*.
- **FTP.** Supports File Transfer Protocol (FTP). This driver is used by Siebel Marketing to send contact lists to vendors who are contracted to execute campaigns. It can also be used to send outbound communication requests from any Siebel application. For more information, see *Siebel Email Administration Guide*.

Communications Drivers and Third-Party Systems

By means of communications drivers, Siebel applications can work with a variety of third-party CTI middleware and email servers. The Siebel communications infrastructure can work with several types of communications systems, in these scenarios:

- Several communications drivers are provided to support Siebel communications modules or third-party communications systems.

- Third-party vendors write communications drivers, using the Adaptive Communications API, to support additional communications systems. (Alternatively, vendors can write a driver to extend a communications driver provided with Siebel Business Applications.)

For updated information about third-party products supported by Oracle or validated to work with Siebel Business Applications, see:

- My Oracle Support Certifications
- Version 8.1.1.x *Siebel Release Notes* on My Oracle Support (Article ID 557191.1)
- Version 8.2.2.x *Siebel Release Notes* on My Oracle Support (Article ID 1050792.1)
- Article ID 477861.1 on My Oracle Support (previously published as Siebel FAQ 2270)

For more information about third-party communications solutions, see the Partners section of the Oracle Web site.

NOTE: For detailed requirements and procedures for installing and configuring third-party products, see your third-party vendor documentation.

The capabilities of your communications system (such as the combination of CTI middleware and switch) partly determine which functionality you can implement. For example, some switches do not support automatic call forwarding. In addition, some features supported in the communications system might not be supported in the corresponding Siebel communications driver. Consequently, information in this guide about certain features might not apply to your implementation. Consult your vendor documentation for information about supported functionality and features. For information about integrating with email servers, see *Siebel Email Administration Guide*.

About Error Messages Related to Communications

For agents using session-based communications such as Siebel CTI, the Siebel client might sometimes receive error messages relating to communications functionality.

Some messages might originate from CTI middleware or from the communications driver. Messages are presented without modification, whether they originate from external communications systems or from Siebel modules. Such messages are prefaced with Communication: and appear at the top of the application area in the browser, to the right of the application-level menus. For more information, see "Using the Menu Commands for Displaying Error Messages" on page 233.

The same error messages might also be logged, if logging is set for the Communications Session Manager component. For more information, see "Parameters for Communications Session Manager" on page 211.

Nonsupported Communications Systems

If your existing communications system is not supported by the communications drivers provided with Siebel Communications Server, then consider the following options for custom communications drivers:

- Use a communications driver developed, using the Siebel Adaptive Communications API, by a third party, such as a CTI middleware vendor. For information about third-party communications solutions validated by Oracle, see the Partners section of the Oracle Web site.

- Develop your own communications driver, using the Siebel Adaptive Communications API, to work with your communications system.

NOTE: Communications drivers developed by customers or third parties are not supported by Oracle, but by their respective developers or providers.

A Siebel Communications Server implementation using a custom communications driver developed using the Siebel Adaptive Communications API can take full advantage of standard features such as the communications toolbar, screen pops, communications commands, automatic creation of activities and logs, and user interface for administration and configuration.

You can optionally create a communications driver to use with an existing driver. Such a driver, known as an aggregate driver, extends the capabilities of the existing driver, enabling additional functionality. For more information, see Appendix A, "Developing a Communications Driver."

Communications Driver Settings

In the Communications Drivers and Profiles view, the Communications Drivers list displays settings for each driver. These settings are discussed in general terms later in this topic.

In most cases, these settings do not have to be changed for the communications drivers provided with Siebel Business Applications. The settings are informative, however, and must be understood by the communications administrator. If you add a custom driver, then you must specify values for these settings.

Driver settings include the following:

- **Channel Type.** For more information, see "Communications Drivers and Channels" on page 36.
- **Inbound.** For more information, see "Inbound and Outbound Drivers" on page 36.
- **Outbound.** For more information, see "Inbound and Outbound Drivers" on page 36.
- **Interactive.** For more information, see "Interactive Drivers" on page 37.
- **Channel String.** For more information, see "Communications Drivers and Channels" on page 36.
- **Library Name.** For more information, see "Communications Driver Files and Database Records" on page 34.
- **Icon File.** For more information, see "Icon Files for Interactive Communications Drivers" on page 37.

Communications Driver Files and Database Records

Communications drivers are usually based on library files, such as DLL (Dynamic Link Library) files on Microsoft Windows, or shared object files on UNIX. Alternatively, they might be executable files or other types of files. Each driver file must be written to support the Siebel Adaptive Communications API.

Communications driver files are located by default in the bin subdirectory of the Siebel Server installation directory for Windows, or in the `lib` subdirectory for UNIX. You can store them in another directory, if you specify an absolute path. (On UNIX, this location cannot be the bin subdirectory of the Siebel Server installation directory.)

Each communications driver file has one or more corresponding database records in the Communications Drivers and Profiles view of the Administration - Communications screen.

NOTE: In this book, in discussions of configuration tasks in the Administration - Communications screen, the term *communications driver* or *driver* sometimes refers to a database record that references a driver file. In discussions of developing drivers, however, the term generally refers to the driver file itself, which is the direct product of the development process.

Driver File Naming on Microsoft Windows and UNIX

The names of actual driver files to load are obtained from the Library Name field in the Communications Drivers list. In some cases, the value for this field might be manipulated to obtain the name of the driver file to load.

Driver File Naming on Microsoft Windows

On Microsoft Windows, the name of the driver file to load is generally obtained by appending .DLL to the value of the Library Name field. However, if the value of the Library Name field includes a period (.), then no manipulation of this value is performed to obtain the name of the driver file.

If you add a new custom driver file that has the extension .DLL, then you can provide a Library Name value in either of two ways:

- The value can correspond exactly to the name of the operating system file. For example, for a file named driver.dll, you can enter driver.dll as the Library Name value. This approach is recommended.
- The value can correspond to the name of the file but without the extension .DLL. For example, for a file named driver.dll, you can enter *driver* as the Library Name value.

Driver File Naming on UNIX

On UNIX, the name of the driver file to load is generally obtained by adding *lib* (without a space character) immediately before the value of the Library Name field and, appending *.so* to the value of the Library Name field. However, if the value of the Library Name field includes a period (.), then no manipulation of this value is performed to obtain the name of the driver file.

If you add a new custom driver file that starts with *lib* and has the extension *.so*, then you can provide a Library Name value in either of two ways:

- The value can correspond exactly to the name of the operating system file. For example, for a file named libdriver.so, you can enter *libdriver.so* as the Library Name value. This approach is recommended.
- The value can correspond to the name of the file but without the elements *lib* and *.so*. For example, for a file named libdriver.so, you can enter *driver* as the Library Name value.

Requirements for Specifying Driver File Names

Some additional requirements for specifying driver file names are as follows:

- If you add a new custom driver file with any other naming pattern, then the file name *must* include a dot, and you *must* provide a Library Name value that corresponds exactly to the name of the operating system file.
- If you install any custom driver file in a nondefault location, then some element of the file or path name *must* include a dot, and you *must* provide a Library Name value that includes the full path where the Siebel Server that is to load the driver can locate the file.
- On UNIX, if you install any custom driver file in a nondefault location, then this location cannot be the `bin` subdirectory of the Siebel Server installation directory. The `lib` subdirectory, the default location for the files provided with the Siebel Business Applications on UNIX, is preferable.
- If you create a custom driver that aggregates an existing Siebel driver, then reference the operating system file for the Siebel driver by its exact name, regardless of what text is displayed for the Siebel driver in the Library Name field.
- If the file path is not specified, then it is assumed that the driver is located in the `bin` subdirectory of the Siebel Server installation directory on Windows, or in the `lib` subdirectory on UNIX. The installation from which the driver file is loaded is the computer on which the channel manager runs.

Communications Drivers and Channels

Developers can write each communications driver file to support a single communications *channel type*, such as voice, email, and so on. Alternatively, developers can write a driver file to support multiple channel types. Most of the Siebel communications drivers each support a single channel type. For example, the Virtual CTI communications driver supports the voice and chat channels.

Each communications driver record includes a string, the *channel string*, that is used by the driver to identify its own channel type. The channel string must match exactly the channel string as defined in the driver file.

For communications driver files that support multiple communications channel types, driver records corresponding to specific channels can reference different subsets of channel-specific functionality within the same driver file, by referencing different channel strings defined within the driver file. For more information, see Appendix A, “Developing a Communications Driver.”

Inbound and Outbound Drivers

Each driver record in the Communications Drivers and Profiles view is indicated to support, or not support, inbound or outbound communications:

- CTI drivers are for both inbound and outbound communications. They also support agent interactivity within the Siebel application.

- The Push Keep Alive driver is an interactive driver that provides a service for agent communications sessions. It does not directly support either inbound or outbound communications.
- Communications drivers for email or fax, such as Internet SMTP/IMAP Server or Internet SMTP/POP3 Server, support both inbound and outbound communications. Inbound functionality for such a driver is applicable only to Siebel Email Response.
- Drivers for paging and FTP support outbound communications only.

For more information about interactive drivers, see "Interactive Drivers" on page 37.

NOTE: The Inbound and Outbound fields are for informational purposes. Do not modify the values for these fields for the drivers provided with Siebel Business Applications.

Interactive Drivers

Depending on the purpose of the driver, the driver record in the Communications Drivers and Profiles view might be indicated as interactive. For example, the Virtual CTI driver and the Push Keep Alive Driver are interactive drivers.

Interactive drivers enable communications functionality to be available to end users using the communications toolbar and related menu controls. Additional capabilities might also be required to route communications work items to an agent's communications toolbar. (The Push Keep Alive driver, unlike other interactive drivers, does not have a user interface component.)

Communications configurations can reference profiles for interactive drivers only, such as the Virtual CTI driver. The events and commands defined in the configuration interact with the communications system through one of the interactive drivers.

NOTE: The Interactive field is for informational purposes. Do not modify the values for this field for the drivers provided with Siebel Business Applications. Custom drivers developed by third parties must be written to be interactive or not interactive, according to their purpose.

Icon Files for Interactive Communications Drivers

Generally, each interactive driver has a specified icon file, which designates the image file that represents the channel for interactive purposes in the communications toolbar. The icon file is used for the Channel Type Indicator and for blinking behavior, such as when an incoming work item has arrived.

The Virtual CTI driver, which supports the voice and chat channels, is an interactive communications driver. For this driver, a default icon file, voice.gif, is specified as part of the driver record data.

For each interactive driver created by third parties, an icon file *must* be specified. Other types of communications drivers do not use icon files. (The Push Keep Alive driver, although it is an interactive driver, does not use an icon file.)

The directory `webmaster\images\`*language_code* in the Siebel Server installation directory stores the source image files, where *language_code* represents the language code for the installed software, such as ENU for U.S. English.

To use your own icon file, place your icon file into the directory identified previously. You can either replace an existing file with your own, or use a new file and specify this filename in the Icon File field for the communications driver.

NOTE: New or updated files on the Siebel Server are automatically populated to the Web server whenever the Web server (where the Siebel Web Server Extension is installed) is restarted. Files can also be manually updated from the Siebel Server without restarting the Web server. For more information, see *Siebel Security Guide* and the *Siebel Installation Guide* for the operating system you are using.

If you are using a custom communications driver (interactive driver), then you can use your own icon file or reuse an existing icon file, such as one that represents the same communications channel.

To work within the existing toolbar framework, each image file used for a driver icon file must be 18 by 18 pixels and must be of format GIF or JPG.

The driver icon files are not the same physical files as the image files that are specified as bitmaps for the toolbar commands in Siebel Tools, although some of the actual images might be identical or closely related. If you modify icons used in one context (interactive icon file or bitmap specified in Siebel Tools), then modify them in both contexts to maintain user interface consistency. For more information about configuring the communications toolbar, see "About Communications Toolbar Configuration" on page 135.

About Profiles for Communications Drivers

A communications driver is initially configured by setting default parameter values for the driver within the Communications Drivers and Profiles view.

A communications driver can be made available to serve a particular purpose by creating one or more profiles that reference this driver. For each profile, parameter values can be specified to override those of the referenced driver or to provide values not otherwise specified.

Communications administrators, who are the primary audience for this book, generally configure drivers and create profiles. Depending on the features or products your company is deploying, multiple administrators might coordinate to perform these tasks.

For example, administrators who have expertise in CTI middleware or email servers might be tasked to configure the communications drivers and profiles that are specific to the technology or product areas with which they are familiar. Or some administrators might configure profiles only, under the direction of another administrator who configures drivers. For more information about creating and using profiles for use with email servers or for outbound communications systems, see *Siebel Email Administration Guide*.

Contexts of Use for Communications Profiles

Communications profiles, whether created by administrators or (in limited contexts) by end users, might be specified for use by administrators and end users in many communications-related contexts within the Siebel application. For more information about creating and using profiles for use with email servers or for outbound communications systems, see *Siebel Email Administration Guide*.

Administrators specify profiles when defining communications configurations to support handling interactive work items such as voice calls for Siebel CTI.

Configuring Communications Drivers and Profiles

This topic describes how to work with communications drivers and profiles to enable Siebel Communications Server to work with communications systems such as CTI middleware, email servers, and so on.

Every Siebel module that uses Communications Server makes use of communications drivers and profiles.

In the Communications Drivers and Profiles view, the Communications Drivers list displays settings for each driver. Generally, the only driver data that you create or modify directly for drivers provided with Siebel Business Applications are the default driver parameter values and the profiles you create for the drivers.

The rest of this topic describes specifying driver parameter values, creating profiles, and adding new driver records for custom driver files created using the Adaptive Communications API.

This topic contains the following information:

- "Specifying Driver Parameter Values" on page 39
- "Field Types for Driver Parameters" on page 40
- "Setting Driver Parameter Default Values" on page 40
- "Defining Communications Profiles" on page 41
- "Specifying Parameter Override Values for Profiles" on page 41
- "Adding a Custom Communications Driver" on page 42

For more information about the parameters supported by each driver provided with Siebel Business Applications, see the topics referenced in "Communications Drivers Provided with Siebel Business Applications" on page 32.

Specifying Driver Parameter Values

For each driver listed in the Communications Drivers and Profiles view, driver parameter data is displayed in the Driver Parameters list. For each driver parameter, a flag indicates whether the parameter requires a value. Driver parameter values can be provided in one of two ways:

- By setting a default value for a driver parameter. For more information, see "Setting Driver Parameter Default Values."
- By specifying an override value for the parameter in each profile you create for the driver. For more information, see "Defining Communications Profiles" on page 41.

Driver parameter values are in effect for all profiles using the driver, unless you override parameter values for a given profile. Values for many parameters are likely to be specific to your site (such as a server name) or to an individual profile (such as a mailbox name).

NOTE: For any required parameter for which no value has been provided as a driver parameter, a value must be provided for the parameter, for each profile, in the form of a profile parameter override. If any parameter is expected to be specified as a profile parameter override, then the driver parameter *must* first be defined in the Driver Parameters list.

Field Types for Driver Parameters

Parameters for communications drivers use one of the following field types:

- Characters (letters, numerals, _ (underscore), and special characters used in macro-expansion)

  Macro expansion can be used in fields of character type, as described in "Using Macro Expansion for Character Fields" on page 151. You can also use macros described in "Macros for Parameter Values" on page 152.

- Boolean (allowable values are True and False)
- Numeric (numerals 0 through 9)

Wildcard characters are not applicable to driver parameter values.

Setting Driver Parameter Default Values

You can specify a new default value for a driver parameter. You can modify any default value in a profile. For certain parameters, you typically provide values through profile parameter overrides.

If you expect that a particular parameter value applies to all or most of the profiles that you create, that is, you do not expect to override the value, then you ought to specify this value as the default driver parameter value. Doing so saves you time when you create profiles, because you might not have to define an override for this driver parameter. For more information about configuring driver parameters for email servers, see *Siebel Email Administration Guide*.

To specify a driver parameter default value

1. Navigate to the Administration - Communications screen, then the Communications Drivers and Profiles view.
2. In the Communications Drivers list, select the driver for which you want to modify the default parameter values.
3. Click the Driver Parameters view tab.
4. In the Driver Parameters list, clear the existing default value for a parameter, and enter a new value.

Defining Communications Profiles

After you configure any applicable communications drivers, as described in "Specifying Driver Parameter Values" on page 39, you create communications profiles for these drivers.

Each profile specifies which driver is to be used and how the driver is used for communications that use the profile.

NOTE: For more information about creating and using profiles with Siebel Email Response, see *Siebel Email Administration Guide*.

To create a communications profile

1. Navigate to the Administration - Communications screen, then the Communications Drivers and Profiles view.
2. In the Communications Drivers list, select the driver for which you are creating a profile.
3. Click the Profiles view tab.
4. In the Profiles list, add a new record to create a profile for the current driver.
5. Provide a name for the new profile.
6. Specify the organization that uses the profile.
7. Specify one or more responsibilities that uses the profile.

Now you specify parameter override values, as described in the following procedure.

Specifying Parameter Override Values for Profiles

This topic describes specifying any new values to override the default values of driver parameters. You must do this for required parameters that are not defined for the driver, or for parameters for which the driver parameter default value is not appropriate for each profile. For more information about driver parameters, see "Specifying Driver Parameter Values" on page 39.

To specify parameter override values for a profile

1. Navigate to the Administration - Communications screen, then the Communications Drivers and Profiles view.
2. In the Communications Drivers list, select the driver for which you created the profile you are configuring.
3. Click the Profiles view tab.
4. In the Profiles list, select the profile you are configuring.
5. In the Profile Parameter Overrides list, add a new record for each driver parameter whose default value you are overriding.
6. For each parameter, specify the parameter name.

7 Enter an override value for the parameter, then commit the record.

   You must specify a parameter override for any parameter for which no value has been provided as a driver parameter.

Adding a Custom Communications Driver

If you have developed a custom communications driver, using the Adaptive Communications API or acquired such a driver from another vendor, then you have to add one or more records for the driver in the Communications Drivers and Profiles view. For information about the driver settings mentioned in this topic, see "Communications Driver Settings" on page 34. For information about developing a custom driver, see Appendix A, "Developing a Communications Driver." See also "Configuring Communications List of Values Types" on page 147.

To add a custom communications driver

1. Navigate to the Administration - Communications screen, then the Communications Drivers and Profiles view.
2. In the Communications Drivers list, add a new record for the driver.

   The Communications Drivers form appears.
3. Provide a name for the driver.
4. Specify the channel type for the driver.
5. Specify the channel string the driver uses to identify its own channel type.
6. Specify if the driver is used for inbound communications.
7. Specify if the driver is used for outbound communications.
8. Specify if the driver is interactive.
9. Enter the name of the file (such as a library file) that this driver record references.

   Include the complete file name of the file, with the file extension.
10. For an interactive driver, enter the name of the driver's icon file.
11. Optionally, provide a description for the new driver.

4 Configuring Siebel CTI

This chapter provides information about configuring Siebel CTI. It includes the following topics:

About Configuring Siebel CTI

This chapter explains how to set up communications configurations and related data. Communications configurations support Siebel CTI and other forms of session communications. This chapter also provides information about importing and exporting communications configuration data.

Related Topics

"Process of Configuring Siebel CTI" on page 24

Chapter 3, "Configuring Communications Drivers and Profiles"

Chapter 5, "Configuring Events and Commands"

Chapter 7, "Configuring Advanced Communications Features"

Chapter 6, "Configuring User Interface Elements"

Chapter 8, "Administering Siebel Communications Server"

Related Books

If you are upgrading from a previous version of Siebel CRM, then see also *Siebel Database Upgrade Guide*.

For all implementations of Siebel CRM, see also related Siebel documentation for your products, such as *Siebel Email Administration Guide*.

Creating or Modifying a Communications Configuration

This topic describes how to create or modify a communications configuration. It contains the following information:

- "Copying or Deleting a Communications Configuration" on page 45
- "Viewing All Communications Configuration Data" on page 46
- "Specifying Parameters for a Communications Configuration" on page 46
- "Parameters for Communications Configurations" on page 47

In the Administration - Communications screen, use the All Configurations view to create or modify a communications configuration.

After you create a communications configuration, you must create and associate parameters, profiles, agents, events, and commands in order for the configuration to be functional.

Create as many separate configurations as you might need in order to group agents who require similar settings for all of the applicable configuration parameters. For an example of a parameter that might have to be set differently for different sets of agents, see the description of the AutoLoadDriver parameter.

TIP: Rather than using the Administration - Communications screen to create or modify communications configuration data directly, you can export some of the data to files and modify or extend it using a text editor. Then you can import the data into the test or production Siebel database when you are ready to test, deploy, or update the communications configuration.

Related Topics

"Specifying Parameters for a Communications Configuration" on page 46

"Parameters for Communications Configurations" on page 47

"Configuring Communications Drivers and Profiles" on page 39

"Specifying Agents" on page 55

"Defining Communications Events" on page 62

"Defining Communications Commands" on page 67

"Exporting and Importing Communications Configuration Data" on page 72

Creating a Communications Configuration

Use the following procedure to create a communications configuration.

To create a communications configuration

1. Navigate to the Administration - Communications screen, then the All Configurations view.

   The All Configurations view appears.

2. In the Configurations list, add a new record.

3 In the Name field, enter the name of the configuration.

4 Add any comments.

5 As appropriate, create or associate elements such as parameters, profiles, agents, commands, and event handlers, as described later in this chapter.

Modify an Existing Communications Configuration

Use the following procedure to modify an existing communications configuration.

To modify an existing communications configuration

1 Navigate to the Administration - Communications screen, then the All Configurations view.

  The All Configurations view appears.

2 In the Configurations list, select the record for the configuration to edit.

3 Make your changes.

4 As appropriate, create or associate elements such as parameters, profiles, agents, commands, and event handlers, as described later in this chapter.

Copying or Deleting a Communications Configuration

You can copy or delete an existing communications configuration record.

This topic is part of "Creating or Modifying a Communications Configuration" on page 44.

Copying a Communications Configuration

You can copy an existing communications configuration record by using the available menu commands.

NOTE: When you copy a record for a communications configuration, associated configuration parameters, profiles, agents, commands (commands and command data), and events (event handlers, event responses, and event logs) are *not* copied.

Deleting a Communications Configuration

You can delete an existing communications configuration by using the available menu commands.

If you delete a record for a communications configuration, then associated configuration parameters, commands (commands and command data), and events (event handlers, event responses, and event logs) are also deleted, recursively. For agents and profiles, however, the associations with these elements are deleted, but the underlying employee records and profile records are *not* deleted. When you perform this operation, the application might take several minutes to respond.

Viewing All Communications Configuration Data

Use the Configuration Explorer view to view information about all the elements of your communications configuration.

The Configuration Explorer view displays communications configuration data in a hierarchical format. This view comprises an explorer (tree control) on the left and lists on the right. The explorer on the left is read-only.

This topic is part of "Creating or Modifying a Communications Configuration" on page 44.

To view communications configuration data

1. Navigate to the Administration - Communications screen, then the Configuration Explorer view.

   The Configuration Explorer view is displayed. Any item in the explorer that is preceded by an arrow might contain other configuration items. For example, configurations contain commands, which in turn contain command parameters and associated command data definitions.

2. Click the arrow for an item in the explorer to expand the item and reveal its specific subitems or categories of subitems.

3. Click an item's name in the explorer applet to display subitem information in the relevant list on the right.

Specifying Parameters for a Communications Configuration

Use the All Configurations view to specify parameter values for a communications configuration. Configuration parameters are in effect for all agents who are included in the configuration for the contact center.

Certain elements associated with a communications configuration also have parameters that affect the overall function of the configuration. Such elements include communications drivers and profiles, commands, and events.

The communications configurations provided with Siebel Business Applications include configuration parameters and default values. "Parameters for Communications Configurations" on page 47 lists these configuration parameters and provides applicable default values, which are in effect for each parameter if it is not defined. Some parameters must be included, while others are optional.

This topic is part of "Creating or Modifying a Communications Configuration" on page 44.

To specify configuration parameters

1. Navigate to the Administration - Communications screen, then the All Configurations view.

2. In the Configurations list, click to select the configuration record for which you are specifying parameters.

3. In the Parameters list:
   a. If the configuration does not include all supported parameters you require, then add a record for each such parameter, specifying the parameter name and parameter value.
   b. Verify or edit values for all the parameters in your configuration.
   c. Click the check mark for the Active field to enable or disable a configuration parameter. A disabled parameter has no effect on the communications configuration.

Parameters for Communications Configurations

This topic describes the configuration parameters for communications configurations and displays applicable default values.

This topic is part of "Creating or Modifying a Communications Configuration" on page 44.

The following are the parameters for communications configurations:

- **AutoLoadDriver.** Specifies the loading behavior of the CTI driver:
  - If AutoLoadDriver is set to True, then the CTI driver is loaded automatically when the user logs in to the Siebel application. If the parameter is not defined, then the default setting is True.
  - You must set AutoLoadDriver to False for an agent to be able to use hoteling if all of the following are true:
    - The agent is using a Siebel application for which Siebel Open UI is enabled.
    - The browser has proxy connections enabled.
    - The agent is logged in to a hoteling computer, that is, a computer whose host name is associated with a teleset located at the same station.

    In this case, after starting the Siebel application, the agent is prompted to enter the local computer's host name in the communications toolbar and then clicks Set Up Hoteling. As a result, the CTI driver is loaded and the agent can then log in to the CTI system or is logged in automatically if automatic login is in effect.

  NOTE: It is strongly recommended that, for the communications configuration in which you set AutoLoadDriver to False, you associate only the agents for which the preceding conditions apply. Otherwise, other agents are prompted to specify the computer host name unnecessarily.

  The ability to set AutoLoadDriver to False is available only for deployments using Siebel Open UI. For more information about hoteling, see "Configuring Telesets for Hoteling" on page 168. For more information about using Siebel Open UI, see *Siebel Fundamentals for Siebel Open UI*.

- **AutoLogin.** Specifies whether automatic login is the global default setting for all agents in the communications configuration, or whether agents can set automatic login:
  - If AutoLogin is set to True, then automatic login is in effect for all agents.
  - If AutoLogin is set to False, then automatic login is disabled for all agents. If the parameter is not defined, then the default setting is False.

- If AutoLogin is set to UserPreference, then whether automatic login is in effect is determined by the automatic login setting (Auto Login to Call Center at Startup) in the agent's User Preferences screen (Communications options).

  If AutoLogin is either True or False, then agents cannot set the preference Auto Login to Call Center at Startup. For more information, see "Preference Settings for Communications" on page 218 and "Configuring Communications Login and Logout" on page 172.

- **AutoLoginCmd.** Specifies which communications command for logging in automatically is executed for each user's session, if AutoLogin is set to True. For more information, see "Preference Settings for Communications" on page 218 and "Configuring Communications Login and Logout" on page 172.
- **AutoLogout.** Specifies whether automatic logout is the global default setting for all agents in the communications configuration, or whether agents can set automatic logout:
  - If AutoLogout is set to True, then automatic logout is in effect for all agents.
  - If AutoLogout is set to False, then automatic logout is disabled for all agents. If the parameter is not defined, then the default setting is False.

  For more information, see "Configuring Communications Login and Logout" on page 172.
- **AutoLogoutCmd.** Specifies which communications command for logging out automatically is executed for each user's session, if AutoLogout is set to True. For more information, see "Configuring Communications Login and Logout" on page 172.
- **BackupCommSessionMgr.** Specifies the name of the backup Communications Session Manager component. The backup component might be running on one or more Siebel Servers. The backup Communications Session Manager component can be accessed without agent interruption, in case the primary Communications Session Manager fails and does not restart.

  Backup functionality for the Communications Session Manager component exists without your configuring any of the related parameters: BackupCommSessionMgr, BackupEnterpriseServer, BackupGatewayAddress, and BackupRequestServer.

  In order to route requests to one or more backup servers for Communications Session Manager, then you can configure these parameters, as needed, to support backup functionality required for different deployment scenarios.

  You *must* define the BackupCommSessionMgr parameter if the backup Communications Session Manager component has a different name than *CommSessionMgr*. Otherwise, it is optional. The value is the component alias (such as CommSessionMgr), not the component name (such as Communications Session Manager).

  For more information about how requests are processed, see 477818.1 (Article ID) on My Oracle Support. This document was formerly published as Siebel Technical Note 570.

  NOTE: It is recommended that you run the backup Communications Session Manager and the Application Object Manager on Siebel Servers for which the Enterprise Servers (one or more) are served by the same Siebel Gateway Name Server. In this case, the BackupEnterpriseServer and BackupGatewayAddress parameters are optional. As noted, the BackupCommSessionMgr parameter is also optional, if the default component alias (CommSessionMgr) is used.

- **BackupEnterpriseServer.** Specifies the name of the Siebel Enterprise for your backup Communications Session Manager components.

  You *must* define this parameter if the Siebel Servers supporting the backup Communications Session Manager and the Application Object Manager run within different Siebel Enterprise Servers. Otherwise, it is optional. For more information, see the description of the BackupCommSessionMgr parameter.

- **BackupGatewayAddress.** Specifies the name of the Siebel Gateway Name Server applicable to your backup Communications Session Manager components. The value is the Name Server computer name or IP address.

  Include the port number if the Siebel Gateway Name Server uses a port other than the default (2320). For example, mygateway:*new_port_num*.

  You must define this parameter if the Siebel Servers with the backup Communications Session Manager and the Siebel Server with the Application Object Manager run within Siebel Enterprise Servers that are served by different Siebel Gateway Name Servers. Otherwise, it is optional. For more information, see the description of the BackupCommSessionMgr parameter.

- **BackupRequestServer.** Specifies the name of a specific Siebel Server on which the backup Communications Session Manager is running.

  By default, backup requests are routed in round-robin fashion to all Siebel Servers on which you are running the Communications Session Manager component. However, if not all of these components use the same communications driver applicable to the current communications configuration, then you can use this parameter to specify a particular server that you know uses the right driver to handle the requests. For more information, see the description of the BackupCommSessionMgr parameter.

- **BringSiebeltoFrontBrowserTitle.** Specifies text to appear in the title bar of the browser window, where the user preference Bring Siebel to Front for a user is set to either On All Incoming Work Items or On Matching Events. If you specify the value *Incoming Interaction*, for example, then this text appears in the user's browser when a new work item arrives.

  The purpose of the parameter is to help the user to notice the new incoming interaction. This parameter is available only for deployments using Siebel Open UI. For more information about Bring Siebel to Front, see "Preference Settings for Communications" on page 218. For more information about Siebel Open UI, see *Configuring Siebel Open UI*.

- **CallFromUICommand.** Specifies the command that is used to make a call to the phone number that a CTI-enabled agent clicks.

  For a phone number field in a list applet, when the agent clicks the hyperlink, a pop-up window appears, in which the agent can click a Call button to make the call to this number. For a phone number field in a form applet, the agent can click an adjacent phone button to make the call.

  This parameter has an effect only if you create command and command data definitions similar to the following examples and then also set this parameter with the value MakeCallToUIPhone (the name of the command that you defined). No call is made if the agent has another current call. If the parameter value is not defined, then no call is made if the agent clicks a phone number.

  ```
  [Command: MakeCallToUIPhone]
  DeviceCommand = "MakeCall"
  Hidden = "True"
  CmdData = "MakeCallToUIPhone"
  ```

  ```
  [CmdData: MakeCallToUIPhone]
  AttachContext = "True"
  Param.CallNotifyText = "Call from {@UserName}..."
  Param.PhoneNumber = "{@UIPhone:Lookup}"
  ```

  For more information about macros such as @UIPhone, see "Macros for Parameter Values" on page 152.

  This parameter is available only for deployments using Siebel Open UI. For more information about Siebel Open UI, see *Configuring Siebel Open UI*. The pop-up window that appears in a list applet and the ability to make a call from a form applet require Siebel Open UI.

- **ChannelCleanupTimer.** Specifies a timeout value, in seconds, that can help the Application Object Manager to clean up orphaned communications sessions, such as in the event of a browser failure.

  If a communications message (for example, a new inbound call) is not successfully pushed from the Communications Session Manager to the Application Object Manager, then the value of ChannelCleanupTimer is compared to the number of seconds since the last successful push message was delivered to the agent's browser. If the number of seconds since the last successful push message was delivered is greater than the value of ChannelCleanupTimer, then the agent's communications session is considered to be orphaned. The session is terminated and communications session resources on the Application Object Manager are freed up for other uses.

  For example, if a message cannot be pushed, then ChannelCleanupTimer is set to 60 (seconds), and if the last successful push message occurred 180 seconds ago, then this communications session is terminated.

  NOTE: It is recommended to define the ChannelCleanupTimer parameter if you use the Push Keep Alive driver with your communications configuration. For more information, see "Using the Push Keep Alive Driver for Session Connections" on page 178.

- **CheckPopupBeforeExecute.** Specifies whether a screen pop is generated only after an agent has closed a pending pop-up window.

  A pop-up window is pending after an agent has initiated an action that displays a pop-up window and before the agent has completed and closed the window. A screen pop is likely to be disruptive to an agent's workflow if it occurs during this time:

  - If CheckPopupBeforeExecute is set to True, then a screen pop is not generated for an agent if a pop-up window is pending. This parameter is True by default.
  - If it is False, then a screen pop might be generated regardless of any pending pop-up.

- **CommSessionMgr.** Specifies the name of the Communications Session Manager component applicable to this communications configuration.

  Define this parameter when the Communications Session Manager is running on a different computer than the Siebel Server on which the Application Object Manager is running.

  You *must* define this parameter if your Communications Session Manager component has a different name than *CommSessionMgr*. Otherwise, it is optional. The value is the component alias (such as CommSessionMgr), not the component name (such as Communications Session Manager).

  You define the CommSessionMgr, RequestServer, EnterpriseServer, and GatewayAddress parameters, as needed. These parameters fully identify the Communications Session Manager applicable to this communications configuration. The RequestServer parameter is always required in order to run Communications Session Manager on a designated computer.

  NOTE: It is recommended that you run Communications Session Manager and the Application Object Manager on Siebel Servers for which the Enterprise Servers (one or more) are served by the same Siebel Gateway Name Server. In this case, the EnterpriseServer and GatewayAddress parameters are optional.

  NOTE: If Siebel Server load balancing is enabled, then you generally run Communications Session Manager on *all* of the load-balanced computers (where Application Object Manager is running). If Communications Session Manager is disabled on *any* of these computers, then you must run it on another computer and specify its location using the parameters described here.

  If you are also running a backup Communications Session Manager, then see also the descriptions for the BackupCommConfigMgr parameter and related parameters. For more information, see "Administering Communications Session Manager" on page 213.

- **ConnectString.** Specifies the connect string to indicate the name of a remote instance of a server such as a CTI middleware server. For more information, see "Configuring Remote Transfers and Conferences" on page 175.

- **CountryCodeDelimiter.** Specify the delimiter to use between a country code and a local area code if an agent makes a call from selected text representing a phone number, such as in the text input field in the communications toolbar. No default delimiter value is defined.

  NOTE: The Virtual CTI driver has a driver parameter named Driver:CountryCodeDelimiter, which has the default value of a period (.). A similar parameter might be defined for a custom driver. Use the same value for the CountryCodeDelimiter configuration parameter as you do for a corresponding driver parameter. For more information, see "Virtual CTI Driver Parameters" on page 236.

- **DialingFilter.Rule*N*.** Used by the Siebel application to manipulate telephone numbers for voice calls made, transferred, or made in a conference call.

  Specifies a set of phone-number translation rules that are invoked when the Lookup or PhoneTypeLookup keyword is specified in macro-expanded text in a communications command for the voice channel (using Siebel CTI). The first set of numbers is searched for. If there is a match, then the searched numbers are translated to the numbers after the -> symbols. For example:

  ```
  DialingFilter.Rule001 = "650506->"
  ```

  This filter rule takes a ten-digit domestic phone number (for example) and translates it into four digits for dialing an internal extension. For more information, see "Working with Dialing Filters" on page 165 and "Using Macro Expansion for Character Fields" on page 151.

- **EnterpriseServer.** Specifies the name of the Siebel Enterprise Server for the Siebel Server on which the applicable Communications Session Manager is running.

  You *must* define this parameter if the Siebel Servers supporting Communications Session Manager and Application Object Manager components run within different Siebel Enterprise Servers. Otherwise, it is optional. For more information, see the description of the CommSessionMgr parameter.

- **GatewayAddress.** Specifies the name of the Siebel Gateway Name Server for the Siebel Server on which the applicable Communications Session Manager is running. The value is the Name Server computer name or IP address.

  Include the port number if the Siebel Gateway Name Server uses a port other than the default (2320). For example, mygateway:*new_port_num*.

  You *must* define this parameter if the Siebel Servers supporting Communications Session Manager and Application Object Manager components run within Siebel Enterprise Servers that are served by different Siebel Gateway Name Servers. Otherwise, it is optional. For more information, see the description of the CommSessionMgr parameter.

- **ImplicitSave.** When set to True, specifies that when an agent has created or modified a record but not yet saved, clicking on the communications toolbar performs an implicit save. When it is set to False (the default), the new or changed record is not saved. (It can be saved or rolled back later.)

- **MaxCommToolbars.** Specifies the number of instances of the Siebel application, for each agent, for which communications session capability can be enabled at one time, that is, for which the communications toolbar can be active.

  If this parameter is not defined, then the applicable value is 1; only one communications toolbar can be active at one time for each agent. It is generally advised to leave this parameter set to 1, so that communications events are received by one Siebel application instance only.

  If an agent is running multiple Siebel application instances (for which communications would otherwise be enabled), then the number of active communications toolbars cannot exceed the value of this parameter. If the maximum has been reached, and another Siebel application instance is started, then the communications toolbar is not displayed in the new instance.

By using the Reset Active Session Count command, an agent can reset communications session capabilities. If the browser stops responding, for example, then the Communications Session Manager might internally retain an agent session that has become unavailable to the agent and might disallow any new agent session that would exceed the value of MaxCommToolbars. The agent can restart the Siebel application and choose Reset Active Session Count, then restart the application again, after which the communications toolbar would be enabled.

To access this command, an agent chooses Tools, then Communications, then Reset Active Session Count from the application-level menu, or uses the keyboard shortcut Ctrl+F8.

The Reset Active Session Count command is configured in Siebel Tools and must not be modified. Optionally, you can modify this shortcut.

The command is available when the user is defined as an agent within a valid configuration and the Enable Communication parameter is True for the agent's application.

- **MessageDisplayInterval.** Specifies a global default setting for the message display interval for all agents in the communications configuration, or enables agents to set this value themselves. The message display interval is the length of time, in seconds, that a message displays in the status bar above the communications toolbar. An example message might be *Call from 6505060000*. For more information, see "Preference Settings for Communications" on page 218. Some example settings for MessageDisplayInterval are as follows:
  - If MessageDisplayInterval is set to UserPreference, then the message display interval is determined by the Message Display Interval setting in the agent's User Preferences screen (Communications options).
  - If MessageDisplayInterval is set to a positive integer value, then this value is in effect for all agents. Agents cannot set the preference Message Display Interval.
  - If MessageDisplayInterval is not set or is set to any other value, then the default message interval of 7 seconds is in effect.
- **MultiTenancy.** Specifies whether or not organization-visibility rules are applied:
  - If MultiTenancy is set to False (the default), then organization-visibility rules are not applied. Use this setting if your Siebel implementation does not use the multitenancy (multiple organization) feature.
  - If the parameter is set to True, then organization-visibility rules apply. Use this setting if your Siebel implementation uses multitenancy.

  For more information, see "Configuring Multitenancy" on page 169.
- **PreferenceLoginCmd.** Specifies which communications command is executed when the agent clicks the Login button in the Agent Queues list, located in the Communications options of the User Preferences screen.

  The default value is PreferenceLoginCmd. For more information, see "Preference Settings for Communications" on page 218 and "Configuring Communications Login and Logout" on page 172.

- **PreferenceLogoutCmd.** Specifies which communications command is executed when the agent clicks the Logout button in the Agent Queues list, located in the Communications options of the User Preferences screen.

  The default value is PreferenceLogoutCmd. For more information, see "Preference Settings for Communications" on page 218 and "Configuring Communications Login and Logout" on page 172.

- **RequestServer.** Specifies the name of the Siebel Server on which the applicable Communications Session Manager is running.

  You *must* define this parameter when the Communications Session Manager is running on a different computer than the Siebel Server on which the Application Object Manager is running.

  For more information, see the description of the CommSessionMgr parameter.

- **RestoreScreenOnWorkResumed.** Specifies whether or not the screen state is restored when a suspended work item of any channel is resumed:

  - If this parameter is set to True, then the screen state is restored when a work item of any channel is resumed. This parameter is True by default.

  - If this parameter is set to False, then the screen state is not restored when a work item of any channel is resumed.

- **RetryBeforeStopPush.** Specifies the number of times that the communications toolbar for a Siebel client retries a request to reconnect with the Siebel Web Server Extension (SWSE) after the client receives an invalid request response (that is, a response to an invalid request) from the SWSE. The default value is 0 (zero): no reconnection requests are sent after an invalid request response is received.

  In some cases, the communications toolbar might continue to send many invalid requests to the SWSE, even after the communications session has been disconnected. However, you can identify invalid request responses by using the StopPushString parameter to specify a string that is part of an invalid request response. When such responses can be identified, then the setting of the RetryBeforeStopPush parameter lets you control whether the communications toolbar continues to send the invalid requests, or how many times it will send them.

  For more information, see the description of the StopPushString parameter.

- **StopPushString.** Specifies a string that identifies an invalid request response (that is, a response to an invalid request sent by the communications toolbar for a Siebel client). The StopPushString parameter value is empty by default; in this case, a request response received is considered invalid if it does not contain the following string:

  ```
  <command name="ClientIP" value="
  ```

  For more information, see the description of the RetryBeforeStopPush parameter.

- **UpdateChannelStatusTable.** Channel status data is logged for display in the All Channel Items view only if the UpdateChannelStatusTable configuration parameter is set to True. By default, this parameter is set to True.

  In a large call center, setting this parameter to True might have performance implications: the Siebel database is updated as communications activity occurs, and the database is queried each time a manager or administrator displays or refreshes the All Channel Items view. For more information, see "Viewing Communications Status Data" on page 197.

Specifying Agents

This topic describes how to specify agents. It contains the following information:

- "Relationship of Agents and Telesets for Siebel CTI" on page 55
- "Agents and Automated Call Distributor Queue Settings" on page 55
- "Specifying Agents for Communications Configurations" on page 56
- "Configuring Agents" on page 57

Use the All Configurations view and the Agent General Profile view to specify and configure agents for your communications configurations.

You specify agents by adding employees to one or more communications configurations in the All Configurations view, as described in "Creating or Modifying a Communications Configuration" on page 44.

As an alternative to adding individual employees to a configuration, you can add employees by adding responsibilities to the configuration. If you do this, then all employees who have the specified responsibilities are added as agents.

After you have added agents, you configure them using the Agent General Profile view. For example, you can associate telesets or specify agent login and password for ACD queues. For more information about the role of agents within a communications configuration, see "About Communications Configuration Data" on page 19.

Relationship of Agents and Telesets for Siebel CTI

If you are using Siebel CTI, then you must associate agents with communications configurations, specify telesets for your call center, and associate the agents with any telesets they use. However, you can choose whether to associate agents with specific hoteling telesets that they use.

NOTE: When an agent starts the Siebel application, any explicit association of the agent with a teleset is overridden if the user's computer (host name) has been associated with the teleset for hoteling purposes.

For integrations using Oracle Contact On Demand and the Virtual CTI driver, you must specify a dummy teleset for each Siebel agent. See also "About the Virtual CTI Driver" on page 235 and related topics. For more information about specifying telesets, see "Specifying Telesets" on page 58. For more information about hoteling, see "Configuring Telesets for Hoteling" on page 168.

This topic is part of "Specifying Agents" on page 55.

Agents and Automated Call Distributor Queue Settings

If agents receive voice calls through ACD queues, then one or more ACD queues can be associated with the agent using the Agent General Profile view.

NOTE: If you are not defining ACD queues, then this topic does not apply.

One or more ACD queues can be designated as primary for the agent. The Log In and Log Out buttons on the communications toolbar log an agent into or out of all primary ACD queues. Automatic login and logout functions can also be used.

In the User Preferences screen, an agent can selectively log in to or out of any of the associated queues, including those not designated as primary. For more information, see "Setting Communications User Preferences" on page 217 and see "Configuring Communications Login and Logout" on page 172. For information about setting ACD queue values, see also "Configuring Communications List of Values Types" on page 147.

This topic is part of "Specifying Agents" on page 55.

Specifying Agents for Communications Configurations

This topic provides instructions for adding agents to a communications configuration. Once you have added an agent to a configuration, using the All Configurations view, you can further configure the agent, including associating the agent to other configurations, using the Agent General Profile view.

This topic is part of "Specifying Agents" on page 55.

To add agents to a communications configuration

1. Navigate to the Administration - Communications screen, then the All Configurations view.
2. In the Configurations list, select the record for the configuration to which you add the agent.

   You can add an agent to more than one configuration, though only one configuration is in effect for the agent at a time.
3. Click the Agents view tab.
4. In the Agents list, add a new record. The Add Agents dialog box appears:
   a. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   b. For each employee you are adding as an agent, click the check box to select this employee record.
   c. Click OK to add all selected employees to the Agents list.
5. Alternatively, in the Agents list, click Add by Responsibilities or choose it from the menu to add to the configuration all agents with the specified responsibilities:
   a. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   b. For each responsibility you are adding, click the check box to select this record.
   c. Click OK to add all selected responsibilities to the Agents list.
6. To add an agent to another configuration, return to Step 2 and repeat the steps that follow.

Configuring Agents

This topic provides instructions for configuring an agent after the agent has been added to at least one communications configuration.

Using the Agent General Profile view, you can add the agent to additional configurations, associate the agent with telesets, and specify the ACD queues from which the agent receives voice calls. You can also designate which configuration is primary for the agent and which ACD queues are primary for the agent.

NOTE: The Agent General Profile view lists only those agents who have been associated with one or more configurations. For more information, see "Creating or Modifying a Communications Configuration" on page 44.

Only one configuration can be designated as primary for the agent, and only one configuration is in effect at one time. When an agent logs in for the first time, the primary configuration is in effect for the agent. In the Communications options of the User Preferences screen, the agent can choose a different configuration, which is in effect the next time the agent logs in. For more information, see "Setting Communications User Preferences" on page 217.

You can associate telesets with an agent in this procedure only if the telesets have been defined. For more information, see "Specifying Telesets" on page 58.

You can define the Agent Login and Password that are retrieved from the database each time an agent logs in to the ACD queue. You must provide these values for agents who are configured to log in to the ACD queue, when required by the ACD. You do not have to specify these values for other users. You only have to specify the agent login and password once for an agent.

NOTE: For an Aspect switch, Agent Login also represents the number that other call-center agents dial to reach the agent.

For more information about configuring login commands, see "Configuring Communications Login and Logout" on page 172. For more information about agent login procedures, see "Using the Communications Toolbar" on page 223.

This topic is part of "Specifying Agents" on page 55.

To configure agents

1. Navigate to the Administration - Communications screen, then the Agent General Profile view.

   The Agent General Profile list displays all agents that have been added to one or more communications configurations.

2. For agents who have to log in to ACD queues, enter the appropriate values into the Agent Login and Password fields, if no values were specified previously.

3. Specify one or more existing configurations to associate this agent with:

   a. Click the Configurations view tab.

   b. In the Configurations list, add a new record. The Add Configurations dialog box appears.

   c. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.

   d. Select one or more configurations with which to associate the agent.
   e. Click OK to associate all selected configurations with the agent.

4. If the agent is associated with multiple configurations, then specify the primary configuration for the agent:
   a. In the Configurations list, select the record of the configuration that is to be the agent's primary configuration.
   b. Click the check box for the Primary field to make this the primary configuration.
5. For voice agents, you can specify one or more existing telesets to associate this agent with:
   a. Click the Telesets view tab.
   b. In the Telesets list, add a new record. The Add Telesets dialog box appears.
   c. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   d. Select one or more telesets with which to associate the agent.
   e. Click OK to associate all selected telesets with the agent.
6. To associate the agent with one or more ACD queues:
   a. Click the ACD Queues view tab.
   b. In the ACD Queues list, add a new record.
   c. Specify an ACD queue the agent uses.
   d. If this ACD queue is one of the primary ACD queues for the agent, then click the check box for the Primary field.

Specifying Telesets

This topic describes how to specify telesets. It contains the following information:

- "Teleset Naming and Hoteling Considerations" on page 59
- "Specifying Telesets, Agents, and Extensions" on page 59
- "Specifying Agents for Telesets" on page 60
- "Specifying Extensions for Telesets" on page 60
- "Viewing Extension Data" on page 61

Use the All Telesets view to specify telesets for your communications configurations, including associating agents and extensions.

Because teleset data is specific to your enterprise, you must specify this data for your call center. Some demo telesets are defined in the Sample Database, but no predefined teleset data is otherwise provided.

Teleset data is indirectly associated with communications configurations, by way of the agents who use the teleset. In this way, Siebel Communications Server can support CTI functions for your call center agents.

You define extensions for the telesets, then associate the extensions with agents (unless you use hoteling) that have already been associated with a configuration. For information about how end users can set preferences for telesets and extensions, see "Setting Communications User Preferences" on page 217.

Teleset Naming and Hoteling Considerations

You can provide any sort of unique name or number for a teleset. Naming telesets after cubicle or station identifiers or computer names might be a good approach to naming telesets. This approach avoids problems that might be associated with naming them after users or extensions, given that multiple users and extensions might be associated with one teleset.

If you implement hoteling, then you might choose to name a hoteling teleset after the host name of the associated hoteling computer.

When you have specified a host name for a teleset in order to support hoteling, any agent logging in to the Siebel client on the hoteling computer also uses the associated hoteling teleset. Associating a host name with a teleset overrides any association of the teleset with an agent. For more information about hoteling, see "Configuring Telesets for Hoteling" on page 168.

This topic is part of "Specifying Telesets" on page 58.

Specifying Telesets, Agents, and Extensions

This topic provides instructions for specifying telesets and specifying agents and extensions for telesets.

This topic is part of "Specifying Telesets" on page 58.

To specify telesets

1. Navigate to the Administration - Communications screen, then the All Telesets view.
2. In the Telesets list, add a new record.
3. In the Teleset field, enter the name of the teleset.
4. If the teleset and computer are to be used for hoteling, then specify in the Host field the name of the computer where the teleset is located.

   The host name you enter is stored using all uppercase letters.
5. Enter any optional comments for the teleset.
6. For a nonhoteling teleset, click the Agents view tab and add agents for the telesets, as described in "Specifying Agents for Telesets" on page 60.
7. Click the Extensions view tab and add extensions for the teleset, as described in "Specifying Agents for Telesets" on page 60.

Specifying Agents for Telesets

This topic provides instructions for specifying agents for telesets. You associate an agent with a teleset in order to authorize the agent to use this teleset.

All agents you associate with telesets must already have been associated with one or more communications configuration. Agents and telesets have a many-to-many relationship. For more information, see "Specifying Agents" on page 55.

Agents who use hoteling exclusively, or who do not use the voice channel (CTI), do not have to be associated with telesets.

This topic is part of "Specifying Telesets" on page 58.

To specify agents for a teleset

1. In the All Telesets view, select the record for the teleset to which you add agents who are authorized users of the teleset.
2. Click the Agents view tab.
3. In the Agents list, add a new record. The Add Agents dialog box appears, displaying employees who have been added as agents to one or more configurations:
   a. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   b. For each agent to associate with the current teleset, click the check box to select this record.
   c. Click OK to add all selected agents to the Agents list.
4. If no Agent Login and Password values exist for a given agent, then you can add this data, as necessary:
   a. Click the hyperlink in the Agent field to drill down to the Agent General Profile view.
   b. In the Agent General Profile list, enter values for the Agent Login and Password fields.

Specifying Extensions for Telesets

This topic describes how to specify extensions for telesets. After you specify telesets, agents, and extensions, the agent who uses the teleset must specify a default standard extension from the teleset that he or she is associated with. The agent does not have to specify a default standard extension if the teleset is a hoteling teleset. For more information, see "Setting Communications User Preferences" on page 217.

This topic is part of "Specifying Telesets" on page 58.

NOTE: Extensions are not required for all environments. For example, for an implementation that uses the Virtual CTI driver, you do not specify extensions.

The way you specify extension data varies according to the switch you use. Table 5 documents how extension type must be specified for each of the following example switches. For more information about supported switches, see My Oracle Support Certifications.

Table 5. Extension Type by Switch

Switch Name	Extension	Extension Type
Avaya (Lucent) Definity G3	Specify one extension for each teleset.	S
Nortel Meridian	Specify two extensions for each teleset: one of each type.	S A
Siemens Hicom 300E (Generic CSTA Phase II)	Specify one extension for each teleset.	S

To specify extensions for a teleset

1. Navigate to the Administration - Communications screen, then the All Telesets view.
2. In the All Telesets view, select the record for the teleset to which you add extensions.
3. Click the Extensions view tab.
4. In the Extensions list, add a new record to create a new extension record.
5. In the Extension field, enter the extension.
6. In the Extension Type field, specify the type, as required for the switch. Specify type S for Standard DN (also called Primary DN) or type A for ACD DN (also called Position DN).
   - For all supported switch types, specify one extension of type S for each teleset.
   - For Nortel Meridian switches only, also specify Extension type A for each teleset.

Viewing Extension Data

Use the All Extensions view to view information about the extensions for your telesets or to locate records for particular extensions.

You do not have to enter data in this view, because it displays data you previously specified in the All Telesets view.

This topic is part of "Specifying Telesets" on page 58.

To view extension data

- Navigate to the Administration - Communications screen, then the All Extensions view.

Defining Communications Events

This topic describes how to define communication events. It contains the following information:

- "Creating Event Logs" on page 62
- "Specifying Event Log Parameters" on page 63
- "Creating Event Responses" on page 63
- "Specifying Event Response Parameters" on page 64
- "Associating Event Logs with an Event Response" on page 64
- "Creating Event Handlers" on page 65
- "Specifying Event Handler Parameters" on page 66
- "Specifying Parameters for the Event Response Associated with an Event Handler" on page 67

You can specify events for your communications configuration. Siebel Communications Server supports three event types:

- Event handlers
- Event responses
- Event logs

In general, you create event logs first, then create event responses and associate event logs with them, then create event handlers and associate event responses with them. You might add parameters for an event response after you create it, and you might also add them after you have associated the event response with a particular event handler. Different procedures are provided for these purposes.

Some event definitions are provided with Siebel Business Applications. Verify whether the events in your communications configuration meet your needs before you modify them or create new events. For more information about the role of events within a communications configuration, see "About Communications Configuration Data" on page 19. For more information about working with communications events, including supported parameters, see Chapter 5, "Configuring Events and Commands."

Creating Event Logs

This topic describes how to create event logs, using the All Event Logs view.

The Event Responses view tab in the All Event Logs view lets you view the event responses with which the current event log is associated.

This topic is part of "Defining Communications Events" on page 62.

NOTE: If you are implementing a custom communications driver, then see also "Configuring Communications List of Values Types" on page 147.

To create an event log

1. Navigate to the Administration - Communications screen, then the All Event Logs view.
2. In the Event Logs list, add a new record.
3. In the Name field, enter the name of the event log.
4. Specify the configuration to associate this event log with:
   a. In the Configuration field, click the select button. The Pick Configuration dialog box appears.
   b. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   c. Select an existing configuration with which to associate the event log, then click OK.
5. Enter any optional comments for the event log.

Specifying Event Log Parameters

This topic describes how to add parameters for an event log.

This topic is part of "Defining Communications Events" on page 62.

To add parameters for an event log

1. Navigate to the Administration - Communications screen, then the All Event Logs view.
2. In the Event Logs list, select the event log for which you want to add parameters.
3. Click the Event Log Parameters view tab.
4. In the Event Log Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type group, choose the parameter name, then type the name of the subparameter after the period. The available event log parameters are documented in "Event Logs" on page 113.
6. In the Value field, enter the value for the parameter.

Creating Event Responses

This topic describes how to create event responses, using the All Event Responses view.

You can associate zero or more event logs with an event response. Some event responses do not need an event log.

The Event Handlers view tab in the All Event Responses view lets you view the event handlers with which the current event response is associated.

The Event Responses view tab in the All Event Logs view lets you view the event responses with which the current event log is associated.

This topic is part of "Defining Communications Events" on page 62.

NOTE: If you are implementing a custom communications driver, then see also "Configuring Communications List of Values Types" on page 147.

To create an event response

1. Navigate to the Administration - Communications screen, then the All Event Responses view.
2. In the Event Responses list, add a new record.
3. In the Name field, enter the name of the event response.
4. Specify the configuration to associate this event response with:
    a. In the Configuration field, click the select button. The Pick Configuration dialog box appears.
    b. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
    c. Select an existing configuration with which to associate the event response, then click OK.
5. Enter any optional comments for the event response.

Specifying Event Response Parameters

This topic describes how to add parameters for an event response.

This topic is part of "Defining Communications Events" on page 62.

To add parameters for an event response

1. Navigate to the Administration - Communications screen, then the All Event Responses view.
2. In the Event Responses list, select the event response for which you want to add parameters.
3. Click the Event Response Parameters view tab.
4. In the Event Response Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

    You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available event response parameters are documented in "Event Responses" on page 100.
6. In the Value field, enter the value for the parameter.

Associating Event Logs with an Event Response

This topic describes how to associate an event log with an event response.

This topic is part of "Defining Communications Events" on page 62.

To associate an event log with an event response

1 Navigate to the Administration - Communications screen, then the All Event Responses view.

2 In the Event Responses view, select the event response with which to associate an event log.

3 Click the Associated Event Logs view tab.

4 In the Associated Event Logs list, add a new record.

5 Specify the event log to associate with this event response:

   a In the Name field, click the select button. The Pick Event Log dialog box appears, displaying existing event logs associated with the same configuration as the event response.

   b Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.

   c Select an event log with which to associate the event response, then click OK.

6 In the Log Type field, specify the type for this log.

   The available types are AddLog, ContextLog, FindLog, Log, MultiLog, and SingleLog. For more information, see the parameter descriptions for these elements in "Event Responses" on page 100.

Creating Event Handlers

This topic describes how to create event handlers, using the All Event Handlers view.

Associating an event response to an event handler is optional. The Event Handlers view tab in the All Event Responses view lets you view the event handlers with which the current event response is associated.

The Associated Event Logs view tab in the All Event Handlers view lets you associate an event log with the event response that is associated with the current event handler. The procedure is the same as that described in "Creating Event Responses" on page 63.

In Step 6 in the following procedure, associating a profile with an event handler is optional. It can help you avoid collisions if the same device event is available in multiple drivers.

This topic is part of "Defining Communications Events" on page 62.

NOTE: If you are implementing a custom communications driver, then see also "Configuring Communications List of Values Types" on page 147.

To create an event handler

1 Navigate to the Administration - Communications screen, then the All Event Handlers view.

2 In the Event Handlers list, add a new record.

3 In the Name field, enter the name of the event handler.

4 Specify the configuration to associate this event handler with:

   a In the Configuration field, click the select button. The Pick Configuration dialog box appears.

   b. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   c. Select an existing configuration with which to associate the event handler, then click OK.

5. Specify the event response to associate this event handler with:
   a. In the Event Response field, click the select button. The Pick Event Response dialog box appears, displaying existing event responses associated with the same configuration as the event handler.
   b. Select an event response with which to associate the event handler, then click OK.
6. Optionally, specify the profile to associate this event handler with:
   a. In the Profile field, click the select button. The Pick Profile dialog box appears, displaying existing profiles associated with the same configuration as the event handler.
   b. Select a profile with which to associate the event handler, then click OK.

      The name of the communications driver for the profile is displayed along with that of the profile.

      NOTE: If you change or remove a profile associated with a configuration, then you must update the values in the Profile field for the configuration's event handlers.
7. In the Device Event field, enter the name of the device event this event handler matches.

   The device events you can specify are those supported by the communications driver for the associated profile, or special events. For information about the events supported by the Virtual CTI driver, see "Virtual CTI Driver Events" on page 253.
8. In the Order field, specify an integer representing the order in which this event handler is checked, relative to other event handlers specifying the same device event.
9. Enter any optional comments for the event handler.

Specifying Event Handler Parameters

This topic describes how to add parameters for an event handler.

This topic is part of "Defining Communications Events" on page 62.

To add parameters for an event handler

1. Navigate to the Administration - Communications screen, then the All Event Handlers view.
2. In the Event Handlers list, select the event handler for which you want to add parameters.
3. Click the Event Handler Parameters view tab.
4. In the Event Handler Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available event handler parameters are documented in "Event Handlers" on page 93.

6 In the Value field, enter the value for the parameter.

Specifying Parameters for the Event Response Associated with an Event Handler

This topic describes how to add parameters for the event response associated with an event handler.

This topic is part of "Defining Communications Events" on page 62.

To add parameters for the event response associated with an event handler

1. Navigate to the Administration - Communications screen, then the All Event Handlers view.
2. In the Event Handlers list, select the event handler for whose associated event response you want to add parameters.
3. Click the Event Response Parameters view tab.
4. In the Event Response Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available event response parameters are documented in "Event Responses" on page 100.
6. In the Value field, enter the value for the parameter.

Defining Communications Commands

This topic describes how to define communication commands. It contains the following information:

- "Creating Command Data Definitions" on page 68
- "Specifying Command Data Parameters" on page 69
- "Creating Commands" on page 69
- "Specifying Subcommands for a Group Command" on page 70
- "Specifying Command Parameters" on page 71
- "Specifying Parameters for the Command Data Definition Associated with a Command" on page 71

You can specify commands for your communications configuration. Siebel Communications Server supports two command types:

- Commands
- Command data definitions

In general, you create command data definitions first, then create commands and associate command data definitions with them. In some contexts, the term *command data* refers to a command data definition. You might add parameters for a command data definition after you create it, and you might also add them after you have associated the command data definition with a particular command. Different procedures are provided for these purposes.

Associating a command data definition to a command is optional. For example, group commands, which contain subcommands, do not need a command data definition. (A group command is a command that includes subcommands.)

Some commands and command data definitions are provided with Siebel Business Applications. Verify whether the commands in your communications configuration meet your needs before you modify them or create new commands.

For more information, see the following:

- For more information about the role of commands within a communications configuration, see "About Communications Configuration Data" on page 19.
- For more information about configuring commands for the communications toolbar and for communications menu items, see Chapter 6, "Configuring User Interface Elements."
- For more information about working with communications commands, including supported parameters, see Chapter 5, "Configuring Events and Commands."

Creating Command Data Definitions

This topic describes how to create command data definitions, using the All Command Data view.

The Commands view tab in the All Command Data view lets you view the commands with which the current command data definition is associated.

This topic is part of "Defining Communications Commands" on page 67.

NOTE: If you are implementing a custom communications driver, then see also "Configuring Communications List of Values Types" on page 147.

To create a command data definition

1. Navigate to the Administration - Communications screen, then the All Command Data view.
2. In the Command Data list, add a new record.
3. In the Name field, enter the name of the command data definition.
4. Specify the configuration to associate this command data definition with:
   a. In the Configuration field, click the select button. The Pick Configuration dialog box appears.
   b. Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.
   c. Select an existing configuration with which to associate the command data definition, then click OK.

5 Enter any optional comments for the command data definition.

Specifying Command Data Parameters

This topic describes how to add parameters for a command data definition.

This topic is part of "Defining Communications Commands" on page 67.

To add parameters for a command data definition

1 Click the Command Data Parameters view tab.

2 In the Command Data Parameters list, add a new record.

3 In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available command data parameters are documented in "Command Data" on page 128.

4 In the Value field, enter the value for the parameter.

Creating Commands

This topic describes how to create commands, using the All Commands view.

The Commands view tab in the All Command Data view lets you view the commands with which the current command data definition is associated.

This topic is part of "Defining Communications Commands" on page 67.

In Step 5 in the following procedure, associating a command data definition to a command is optional. For example, group commands, which contain subcommands, do not need a command data definition. (A group command is a command that includes subcommands.)

In Step 6 in the following procedure, associating a profile with a command is optional. It can help you avoid collisions if the same device command is available in multiple drivers.

NOTE: If you are implementing a custom communications driver, then see also "Configuring Communications List of Values Types" on page 147.

To create commands

1 Navigate to the Administration - Communications screen, then the All Commands view.

2 In the Commands list, add a new record.

3 In the Name field, enter the name of the command.

4 Specify the configuration to associate this command with:

   a In the Configuration field, click the select button to display the Pick Configuration dialog box.

   b Select an existing configuration to associate with the command, then click OK.

5 Optionally, specify the command data to associate with this command:

   a In the Command Data field, click the select button. The Pick Command Data dialog box appears, displaying existing command data definitions associated with the same configuration as the command.

   b Scroll to display any additional records that are not shown, or use Query or Find to locate records matching certain criteria.

   c Select a command data definition to associate with the command, then click OK.

6 Optionally, specify the profile to associate this command with:

   a In the Profile field, click the select button. The Pick Profile dialog box appears, displaying existing profiles associated with the same configuration as the command.

   b Select a profile to associate with the command, then click OK.

   The name of the communications driver for the profile is displayed along with that of the profile.

7 Enter any optional comments for the command.

Specifying Subcommands for a Group Command

This topic describes how to specify subcommands for a group command. (A group command is a command that includes subcommands.) For more information about group commands that specify subcommands, see "Communications Group Commands in the Communications Toolbar" on page 144.

This topic is part of "Defining Communications Commands" on page 67.

NOTE: When you specify subcommands for a group command, do not specify invalid recursive relationships between subcommands and group commands. For example, do not specify a subcommand that is itself a group command for which the current group command is specified as a subcommand. A relationship like this invalidates the communications configuration when it is loaded for an agent's session and disables the communications toolbar.

To specify subcommands for a group command

1 Navigate to the Administration - Communications screen, then the All Commands view.

2 In the Commands list, create or select a command to which you add subcommands.

3 Click the Subcommands view tab.

4 In the Subcommands list, add a new record.

5 In the Subcommands field, click the select button to display the Pick Command dialog box.

6 Select an existing command to serve as a subcommand for the current command, then click OK.

7 Specify the order for the subcommand.

8 Repeat Step 4 through Step 7, as necessary, for each subcommand.

Specifying Command Parameters

This topic describes how to add parameters for a command.

This topic is part of "Defining Communications Commands" on page 67.

To add parameters for a command

1. Navigate to the Administration - Communications screen, then the All Commands view.
2. In the Commands list, select a command to which you add parameters.
3. Click the Command Parameters view tab.
4. In the Command Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available command parameters are documented in "Commands" on page 116.

   For example, the DeviceCommand parameter lets you specify a device command to execute, such as for a communications driver to pass on to an external communications package such as CTI middleware.
6. In the Value field, enter the value for the parameter.

Specifying Parameters for the Command Data Definition Associated with a Command

This topic describes how to add parameters for the command data definition associated with a command.

This topic is part of "Defining Communications Commands" on page 67.

To add parameters for the command data definition associated with a command

1. Navigate to the Administration - Communications screen, then the All Commands view.
2. In the Commands list, select a command for which you are adding parameters to the associated command data definition.
3. Click the Command Data Parameters view tab.
4. In the Command Data Parameters list, add a new record.
5. In the Name field, specify the name of the parameter.

   You can choose the parameter name from a list or type the name. For a parameter of type Group, choose the parameter name, then type the name of the subparameter after the period. The available command data parameters are documented in "Command Data" on page 128.
6. In the Value field, enter the value for the parameter.

Exporting and Importing Communications Configuration Data

This topic describes how to export and import configuration data. It contains the following information:

- "Exporting Communications Configuration Data" on page 73
- "Importing Communications Configuration Data" on page 74
- "Communications DEF Files" on page 76
- "Example Section from a DEF File" on page 76

Siebel Communications Server provides import and export mechanisms that enable administrators to move communications configuration data between the database and ASCII text files. You can use these mechanisms to:

- Move configuration data between databases or between communications configurations within the same database
- Move configuration data and custom communications drivers obtained from another source such as a third-party vendor into Siebel Communications Server
- Edit or enter data in text files rather than directly within the Administration - Communications screen

For more information about adding custom communications drivers, see Chapter 3, "Configuring Communications Drivers and Profiles."

The basic options for exporting and importing are as follows:

- Export communications configuration data from the database to a DEF file
- Import communications configuration data into the database from a DEF file (or from an INI file exported from a previous release)

  Table 6 describes how driver and profile parameters are handled by the import process from a DEF file to the database.

Table 6. Parameter Management by the Import Process

If the parameter ...	For example ...
Does not exist in the database, then the import process adds it.	If the DEF file contains an entry: `Driver:MessageInterval = "1000"` then the import process updates the database as follows: `Driver:MessageInterval = "1000"`

Table 6. Parameter Management by the Import Process

If the parameter ...	For example ...
Exists in the database, then the import process updates it.	If the database contains an entry: `MessageInterval = "500"` and the DEF file contains an entry: `Driver:MessageInterval = "1000"` then the import process updates the database as follows: `Driver:MessageInterval = "1000"`
Is deleted from the DEF file, then it remains in the database after the import process is completed.	If both the DEF file and the database contain the entry: `Driver:MessageInterval = "1000"` but then you delete the entry from the DEF file and run the import process, then the database retains the entry that you deleted from the DEF file.

For more information about the DEF file format, see "Communications DEF Files" on page 76.

NOTE: Using the Export Configuration and Import Configuration features, you can transfer all types of communications configuration data from one configuration to another *except* data for agents (and ACD queues) and telesets (and extensions).

Exporting Communications Configuration Data

Exporting communications configuration data from the database into a file can make it easier to review your configuration settings. You might want to do this before you put your system into production or before you consult with Global Customer Support. Or, if you would prefer to modify communications configuration data in text files, then you might export and import such data multiple times to test your configuration changes.

For help with defining communications configurations, create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

To export communications configuration data

1. Navigate to the Administration - Communications screen, then the All Configurations view.
2. In the Configurations list, select the configuration record for which to export data.

3 Click Export Configuration, located at the top of the view.

  A dialog box appears that contains four export options:

  - **Configuration Parameters.** Exports configuration parameters. This data is displayed in the All Configurations view, in the Configuration Parameters list.
  - **Commands.** Exports commands and command data. This data is displayed in the All Commands and All Command Data views.
  - **Events.** Exports event handlers, event responses, and event logs. This data is displayed in the All Event Handlers, All Event Responses, and All Event Logs views.
  - **Drivers and Profiles.** Exports communications drivers and profiles. This data is displayed in the Communications Drivers and Profiles view.

4 Click the applicable check boxes named in Step 3 to specify what kinds of data to export.

  The following message is displayed: *Please press OK button to begin exporting. The process may take several minutes. Please wait while exporting...*

5 Click OK.

  You are prompted whether to open or save this file.

6 Specify whether to open or save the file, then click OK:

  - If you choose to open the file, then the DEF file is opened in your default text editor.
  - If you choose to save the file, then you are prompted to choose a location and file name to save to. Click Save to export to this destination file.

  NOTE: When you export a configuration, the prompted filename has the extension .DEF. However, files when exported might be appended by .TXT or another extension, depending on your computing environment.

7 After the export has completed, close the Export Configuration dialog box.

8 Rename the exported file to change the extension to .DEF.

Importing Communications Configuration Data

You might have to import communications configuration data from a file into the database for several reasons. You can import data previously exported from the same Siebel CRM version or from a previous version. You can also use DEF files to import data that you previously exported from another configuration or another Siebel database, for example, exporting data from a test Siebel database and importing it into your production Siebel database.

An import file must contain valid data and use correct DEF file notation. The import file must be consistent with files exported from the current Siebel CRM version, or consistent with previous versions. Each configuration file that you export has an entry like this:

```
[Siebel]
    CommServerVersion = "7.0"
```

CAUTION: The preceding entry applies to any Siebel CRM version 7.x or 8.x software. For any communications configuration file that you import that does *not* contain an entry as shown, upgrade steps are performed on the file. Do not manually add such an entry to a configuration file unless the file is known to be valid and does not have to be upgraded.

For files that were created in previous Siebel CRM versions (before version 7.0), the upgrade scenario applies. When importing files from previous versions, import the INI file first, and then the DEF file.

For more information about the DEF file format, see "Communications DEF Files" on page 76.

CAUTION: When you import communications configuration data, all existing data of the same type is first deleted from the database for the current configuration. New data is then inserted. You cannot cancel an import operation after you begin it. If you are not sure of the effects of importing configuration data, then export the old configuration data to a DEF file before you import the configuration data.

To import communications configuration data

1. Navigate to the Administration - Communications screen, then the All Configurations view.
2. In the Configurations list, create a new configuration record, or select the configuration record for which to import data.

   For information about creating configurations, see "Creating or Modifying a Communications Configuration" on page 44.
3. Click Import Configuration, located at the top of the view.
4. Click Next.

   A dialog box appears that contains four import options:

   - **Configuration Parameters.** Imports configuration parameters. This data is displayed in the All Configurations view, in the Configuration Parameters list.
   - **Commands.** Imports commands and command data. This data is displayed in the All Commands and All Command Data views.
   - **Events.** Imports event handlers, event responses, and event logs. This data is displayed in the All Event Handlers, All Event Responses, and All Event Logs views.
   - **Drivers and Profiles.** Imports communications drivers and profiles. This data is displayed in the Communications Drivers and Profiles view.
5. Click the applicable check boxes named in Step 4 to specify what kinds of data to import.
6. Enter the name of the source file, or click Browse and select the file.
7. Click OK to import the specified configuration data from the source file.

   Wait for the import process to finish, as shown by the imported configuration elements displayed in the Siebel application user interface. After the import has completed, the dialog box closes.

Communications DEF Files

Communications DEF files are files that contain communications definitions corresponding to data of the type that can be defined in a communications configuration. These text files employ the following file format:

- The names in brackets ([ ]) identify elements such as drivers, profiles, commands, and events. Each bracketed name is followed by lines containing names and values for parameters.
- Field values are enclosed in double quotation marks (which are not used for this data within the Administration - Communications screen) and are indicated using an equals sign (=), as in the following example line from a communications command:

  ```
  DeviceCommand = "MakeCall"
  ```

- Parameters are used in DEF files to represent some elements or relationships that you do *not* specify by using parameters when you work in the Administration - Communications screen. For example, the following line, for an event handler definition, associates the event handler with an event response:

  ```
  Response = "InboundConsumerCall"
  ```

- Any lines preceded by a semicolon (;) are not in effect and might contain explanatory comments.
- A DEF file that you export contains lines like the following:

  ```
  [Siebel]
      CommServerVersion = "7.0"
  ```

  Do not remove these lines from your export files. See also "Importing Communications Configuration Data" on page 74.

If you have to manually prepare a file for import, then you can export a file of the same type containing preconfigured data, then check that your file to be imported uses the correct format.

Example Section from a DEF File

Each communications DEF file includes definitions like the ones in the following example from a configuration that supports the Virtual CTI driver.

```
[EventHandler:InsideCallFromCOD]
Filter.CallConnected      = "1"
Filter.InboundCall        = "1"
Filter.SecondaryOperation = "1"
Filter.SiebelCall         = "1"
DeviceEvent               = "InteractionMessage"
Response                  = "InsideCallFromCOD"
Order                      = "2"

[EventResponse:InsideCallFromCOD]
UseCtxData = "True"
Log        = "LogIncomingCallContactFound"
```

```
[EventLog:LogIncomingCallContactFound]
AfterWork.'ACD Call Duration'  = "{@WorkDuration}"
AfterWork.'Done'               = "{@Now}"
AfterWork.'Planned Completion' = "{@Now}"
BusComp                        = "Action"
BusObj                         = "Contact"
Display                        = "False"
LogField.'Account Id           = "{Contact.'Account Id'}"
LogField.'Call Id'             = "{InteractionId}"
LogField.'Contact Id (Thin)'   = "{Contact.Id}"
LogField.'Display'             = "Activities Only"
LogField.'Planned'             = "{@WorkStartTime}"
LogField.'Started'             = "{@WorkStartTime}"
LogField.Description           = "Inbound call"
LogField.Type                  = "Call - Inbound"
ServiceMethod                   = "Persistent Customer Dashboard.Update Dashboard
from CTI"
ServiceParam.Field             = "Id"
ServiceParam.Value             = "{Contact.Id}"
WorkTrackingObj.ContactId       = "{Contact.Id}"
```

5 Configuring Events and Commands

This chapter provides information about configuring events and commands for session communications, such as for Siebel CTI, to serve your business needs. It includes the following topics:

About Events and Commands

This topic provides information about events and commands. It contains the following information:

You can customize the way Siebel Communications Server operates for functions supported by the Communications Session Manager server component:

NOTE: Communications events and commands apply only to communications involving interactive drivers that control communications toolbar functionality, such as for the voice channel.

- You can customize the way Communications Server handles *events* received from the communications system, such as your CTI middleware. Examples of events are when an inbound work item such as a voice call or email message is routed to an agent, when the agent accepts an inbound work item, when an agent completes a conference call, and so on.
- You can customize *commands* that are to be sent from Communications Server to the external communications system, or that are to invoke some other function. Examples of commands are when an agent accepts an inbound work item, initiates an outbound work item, transfers a work item, and so on.

You can customize event or command definitions, for example, to attach different types of data to incoming, outgoing, or transferred communications work items.

For an overview of communications configuration data, see "About Communications Configuration Data" on page 19.

For information about how to use views in the Administration - Communications screen to modify events and commands, see "Defining Communications Commands" on page 67 and "Defining Communications Events" on page 62.

Communications Definition Data in the Database

The Siebel database contains configuration data that helps determine how your communications system is integrated with your Siebel application. Among other functions, the communications configuration data:

- Defines how a Siebel application handles events received from the communications system
- Defines how commands are generated and sent from the Siebel application to the communications system
- Defines the appearance and availability of commands in the Communications submenu in the Siebel application, by specifying hot keys, menu command titles, menu item sequence, and so on

NOTE: The events recognized by the communications driver and the commands supported by the driver are a subset of those supported by the communications system, such as your CTI middleware. For information about the events, commands, and parameters supported by the Virtual CTI driver, see Chapter 10, "Using the Virtual CTI Driver."

To customize the communications configuration, you must understand your call center's workflow model, as well as the key business objects for your Siebel application.

You customize communications features by editing or entering data in a series of views in the Administration - Communications screen, as described in Appendix C, "Views for Communications Administration."

Communications Data Sets

A communications *data set* defines the commands, events, and associated data passing between the elements of your communications system, such as between your CTI middleware and the communications-enabled Siebel application that an agent is running. Events and commands have attributes and corresponding data.

Data set attributes can be supplied by:

- External modules such as call control tables, campaign manager modules, predictive dialers, voice response units, call routers, and so on.
- Siebel features or modules that play a role in supporting communications received from external systems, for example, Siebel CTI, Siebel Email Response, Siebel Workflow, Siebel business service methods, and so on.

Each attribute has a name and contains associated data, such as the telephone number for the caller or for the caller's organization.

To see examples of communications data sets, you can view the Siebel Communications Server log file for the communications drivers you use and for your external communications systems.

Each type of communications event generated has a unique name and an associated data set. Each event can be handled individually by the communications system. Similarly, commands the Siebel client sends to the communications system can accept parameter strings and data sets.

For example, a command to transfer a work item can, along with parameter data specifying data such as a phone extension, accept any data set. When transferring the work item, the data set might include a service request ID or context information for the agent's current Siebel view.

Communications Server associates the data set with the work item (*attaches data*) and sends the data set to the Siebel client of the agent receiving the transferred work item. A screen pop might be generated for the receiving agent, causing the current Siebel view to be displayed with a relevant record, such as a customer's current service request. For more information about the relationships among the various types of communications configuration data, see "Types of Communications Configuration Data" on page 19.

Event and Command Definitions

This topic lists the types of events and commands supported by Siebel Communications Server, describes the field types for event and command parameters, describes how wildcard characters can be used in parameter values, and notes the form in which parameters are defined in events or commands.

Communications configuration parameters, communications drivers and profiles control the feature set provided by the communications system. Event and command definitions control how and when these features are used, how data attached to communications work items affects application behavior, and so on. Each event or command definition must have a unique name within its type.

Each communications command definition or event definition has a unique name within its type and includes a set of parameters for which values are specified. Each type of event or command, and its supported parameters, is described in detail later in this chapter.

Event and Command Types

The following types of events and commands are defined in a communications configuration:

- **Event.** Communications Server supports three types of event definitions:
  - **Event handler.** Defines how the Siebel client application responds to events in the communications system. Specifies a response to execute when an event handler matches an event. For more information, see "Event Handlers" on page 93.
  - **Event response.** Determines a specific response to an event that matched the event handler that invokes the response. For more information, see "Event Responses" on page 100.

  - **Event log.** Defines Siebel log-generation rules for communications events. For instance, events might be logged as activities (Action business component) or logged as another type of record. For more information, see "Event Logs" on page 113.

- **Command.** Defines communications commands that can be invoked in the Siebel application and specifies how they are to appear in the application. For more information, see "Commands" on page 116.

- **Command data.** Defines data associated with a communications command that is invoked. For more information, see "Command Data" on page 128.

Event and Command Parameters

This topic provides general information about parameters you can define for your event and command definitions.

Field Types for Event and Command Parameters

Parameters for each type of event or command use one of the following field types:

- Characters (letters, numerals, _ (underscore), and special characters used in macro-expansion)

  Macro expansion can be used in fields of character type, as described in "Using Macro Expansion for Character Fields" on page 151. You can use the macros described in "Macros for Parameter Values" on page 152.

- Boolean (allowable values are True and False)

- Numeric (numerals 0 through 9)

Forms of Event and Command Parameters

Parameters of character type can be one of the following forms:

- Single parameter

  The parameter must be unique within the definition.

- Multivalue

  The parameter name does not have to be unique within the definition. Multiple instances of a multivalue parameter can be defined; each instance has a different value.

- Group of subparameters

  The group parameter name must be unique within the definition, but multiple parameter instances can be defined, each specifying a unique subparameter. For instance, if you have parameters called Filter.A and Filter.B, then A and B are subparameters of the group parameter Filter.

Table 7 illustrates the usage of parameters of the three forms of character fields. Parameters must be specified in a form similar to what is shown in this table.

Table 7. Event and Command Parameter Forms

Parameter Name	Parameter Value
SingleParam	Value
MultiValueParam	Value1
MultiValueParam	Value2
GroupParam.SubParam1	Value
GroupParam.SubParam2	Value

Wildcard Characters in Event and Command Parameters

The wildcard characters asterisk (*) and question mark (?) can be used within communications event or command parameter values of character type.

Use the wildcard characters to perform pattern matching on communications event data field values or on Siebel database field values. Such matching can help determine which event or command is to be invoked.

Separately, asterisk (*) matches zero or more characters, and question mark (?) matches exactly one character. You can use question mark and asterisk together (?*) to match field values when the corresponding field value must not be empty. This combination of wildcard characters matches one or more characters.

NOTE: The characters asterisk (*) and question mark (?) do not function as wildcards for event or command parameters that pass parameters to a Siebel business service, a Siebel SmartScript, or a Siebel VB or Siebel eScript script.

Special Events for Device Events

This topic describes special events, which are available for any communications configuration and are independent of the communications driver. Special events can be specified as device events in event handlers.

This topic contains the following information:

- "Special Event Attributes" on page 84
- "List of Special Events" on page 85

For more information about device events, see "Defining Communications Events" on page 62, and see the description for the DeviceEvent parameter in "Event Handler Parameters" on page 95. For more information about defining event handlers, see "Event Handlers" on page 93.

The special events are not received from the communications system, such as CTI middleware, as are many other communications events for which a device event is specified.

NOTE: Special events are primarily used for defining event handlers in your communications configuration. However, you can also invoke them from scripts written using Siebel VB or Siebel eScript.

When using special events, you use WorkItemID, instead of SiebelWorkItemID, to refer to work item attributes. For example, you might define an event handler as follows:

```
[EventHandler:OnWorkItemStarted]
   DeviceEvent = "@PreWorkItemStartedEvent"
   FilterSpec = "[$GetWorkItemAttr(WorkItemID, ANI)] IS NOT NULL"
```

Many of the special events correspond to client handle methods and include work item attributes. For more information, see "Work Item Attributes" on page 160. For more information about the client handle methods, see "Methods of ISC_CLIENT_HANDLE" on page 273.

Special Event Attributes

Many of the special events shown in Table 8 on page 85 have some or all of the following attributes:

- **ChannelType.** The language-independent value representing the channel type of this work item.
- **ProfileName.** The communications driver profile for which this work item is applicable.
- **WorkItemID.** The ID number of this work item.
- **WorkItemMode.** The mode of this work item, where 1 represents an inbound work item, and 2 represents an outbound work item.

List of Special Events

The special events, which have names that start with the symbol @, are described in Table 8.

Table 8. Special Events

Event Name	Description
@PreIndicateNewWorkItemEvent @PostIndicateNewWorkItemEvent	Respectively, these special events correspond to just before, and just after, the client handle method IndicateNewWorkItem is invoked. These special events have the following attributes: ■ ChannelType ■ ProfileName ■ WorkItemID ■ WorkItemMode
@PreWorkItemReleasedEvent @PostWorkItemReleasedEvent	Respectively, these special events correspond to just before, and just after, the client handle method WorkItemReleased is invoked. These special events have the following attributes: ■ ChannelType ■ ProfileName ■ WorkItemID
@PreWorkItemResumedEvent @PostWorkItemResumedEvent	Respectively, these special events correspond to just before, and just after, the client handle method WorkItemResumed is invoked. These special events have the following attributes: ■ ChannelType ■ ProfileName ■ WorkItemID
@PreWorkItemStartedEvent @PostWorkItemStartedEvent	Respectively, these special events correspond to just before, and just after, the client handle method WorkItemStarted is invoked. These special events have the following attributes: ■ ChannelType ■ ProfileName ■ WorkItemID

Table 8. Special Events

Event Name	Description
@PreWorkItemSuspendedEvent @PostWorkItemSuspendedEvent	Respectively, these special events correspond to when the client handle method WorkItemSuspended is invoked. These special events have the following attributes: ■ ChannelType ■ ProfileName ■ WorkItemID
@RuntimeEvent	Corresponds to a Siebel run-time event, which was defined in the Administration - Runtime Events screen. This special event has the following attributes: ■ **Context.** The business service context. ■ **ActionSet.** The name of the action set. ■ **Action.** The name of the action. ■ **EventId.** The ID of the run-time event. ■ **Event Name.** The name of the run-time event (for example, *SetFieldvalue*). ■ **Sub Event.** The name of the subevent. ■ **Event Type.** The event type specified in the Object Type field. ■ **Object Name.** The object name. ■ **BusObjName.** The business object name. ■ **BusCompName.** The business component name. ■ **ActiveRowID.** The current active row ID of the business component. ■ **ViewName.** The current view name. For more information about configuring run-time events, see *Siebel Personalization Administration Guide*.

Special Commands for Device Commands

This topic describes special commands, which are available for any communications configuration and are independent of the communications driver. Special commands can be specified as device commands in commands.

This topic contains the following information:

- "List of Special Commands" on page 87
- "Special Command Parameters" on page 92
- "Examples of Special Commands" on page 92

For more information about device commands, see "Defining Communications Commands" on page 67 and see the description for the DeviceCommand parameter in "Command Parameters" on page 119.

For more information about defining commands, see "Commands" on page 116.

The communications driver does not receive these commands and so does not pass them to a system such as CTI middleware, as with many other device commands you specify. Some special commands might, however, require another module.

NOTE: Special commands are primarily used for defining commands in your communications configuration. However, you can also invoke them from scripts written using Siebel VB or Siebel eScript.

Each special command allows attachment of *any* user-defined key or value pair to the work item. Within the communications configuration, these key or value pairs are represented by event or command parameters and the associated parameter values.

Commands in which special commands are specified as device commands support the event and command parameters documented in Chapter 5, "Configuring Events and Commands." In your command definitions, you can specify custom subparameters for command parameters of type Group.

List of Special Commands

The special commands, which have names that start with the at sign (@), are described in Table 9 on page 88. In this table, an asterisk (*) before a parameter name means that the parameter is optional for this command. Parameters for these special commands are described in Table 10 on page 92.

Table 9. Special Commands

Command Name	Parameters	Description
@Associate	*bus_comp_field*	Updates a work item’s tracking object. This object is specified using the event logging feature and the AfterWork event log parameter. **NOTE:** In order for a command to be enabled that is based on the @Associate special command, all applicable event log definitions must include the AfterWork parameter. AfterWork.'ACD Call Duration' = "{@WorkDuration}" For more information about the AfterWork parameter, see “Event Logs” on page 113. To associate the tracking object to an account record, for example, the associated command data definition includes parameters like these, to specify the business component and field: BusComp = "Account" Param."Account Id" = "{Id}" Parameter names enclosed in single or double quotes, as shown for Param."Account Id", are interpreted as field names. Values for such a parameter (enclosed in quotes on the right) are interpreted as field values, for which macro-expansion is supported. (The quotes enclosing the parameter values in this example are *not* part of the parameter values.) The @Associate command always operates on the selected work item: the work item with the current focus, which an agent has selected from the Work Items list in the communications toolbar. For more information about event logging, see “Event Logs” on page 113.

Table 9. Special Commands

Command Name	Parameters	Description
@CreatePopupFrame	*AppletMode *Dimension *PageURL	Causes the browser to display a window in which to display content. The browser window content can be filled in as appropriate for a specified URL, based on what you specify for the PageURL parameter. Alternatively, it can display an applet. For example, a command definition using @CreatePopupFrame as the device command might have an associated command data definition that provides a URL to be displayed in the window, for example: Param.PageURL = "http://www.example.com" The URL can also be specified without the prefix *http://*, if the URL begins with *www.* A third-party company integrating with Siebel applications can use this method to open a new browser window and connect to its own Web server for integration purposes. In order to display an applet, the command data for a command using @CreatePopupFrame must include the following parameters: Param.AppletMode = "*mode_name*" SelectParam = "True" SelectApplet = "*applet_name*" where *mode_name* is either Edit or Base (use Edit for form applets and Base for list applets) and *applet_name* is the applet name in Siebel Tools. Base is the default value for AppletMode. The browser window dimensions can be specified using the Dimension parameter. For example, the following parameter creates a window with dimensions 500 by 300 pixels: Param.Dimension = "500x300" If the Dimension parameter is not defined, then the dimensions are determined by the applet specified using SelectApplet, or by the browser.

Table 9. Special Commands

Command Name	Parameters	Description
@InvokeSWECommand	SWECommand	Invokes a command on the Siebel Web Engine (SWE). The SWE command is specified using the SWECommand parameter. For example, for the command SendFaxGroup, which uses @InvokeSWECommand as the device command, this command data parameter is defined: Param.SWECommand = "Send Fax (SWE)" All of the Send commands (Send Email, Send Fax, Send Wireless Message, and Send Page) are invoked using @InvokeSWECommand.
@OpenView	View	Navigates the Siebel application to the specified view. In the command data definition, specify the view name as defined in Siebel Tools, for example: Param.View = "Service Contact Detail View"
@UpdateRecord	*bus_comp_field*	Updates a database record for the current business component. For example, if the current view is based on the Service Request business component, then you can change a status-related field for this business component. In the command data definition, specify parameters based on what business component field to update and what values the field supports. For example: BusComp = "Service Request" Param.'Status' = "Pending"

Table 9. Special Commands

Command Name	Parameters	Description
@ViewWorkObject	View	Views a work-tracking object for a work item. Navigates to the view, such as a view in the Activities screen, where you can view the tracking object's record. @ViewWorkObject can be invoked from a business service, but not directly from a script. **NOTE:** In order for a command to be enabled that is based on the @ViewWorkObject special command, all applicable event log definitions must include the AfterWork parameter. AfterWork.'ACD Call Duration' = "{@WorkDuration}" For more information about the AfterWork parameter, see "Event Logs" on page 113. A command data parameter with the name Param.View must exist and must contain the name of the view in which to view the tracking object. For example, if the tracking object is Action (for activities), then this record can be found in the view specified as follows: Param.View = "Activity Attachment View" **NOTE:** The view must be based on the same business component specified in the event log. The @ViewWorkObject command always operates on the selected work item: the work item with the current focus, which an agent has selected from the Work Items list in the communications toolbar. **NOTE:** Work-item duration is updated for the work-tracking object (such as an activity record) when a work item is released. This update might conflict with changes made by the agent. Consequently, an agent viewing a work-tracking record must save any changes before releasing the work item. To make changes after releasing the work item, the record must first be refreshed.

Special Command Parameters

Table 10 lists special command parameters and their usage. Table 9 on page 88 identifies the commands that use each of these parameters. Within the communications configuration, special command parameters and values can be specified within command data definitions as subparameters of the Param parameter. For more information, see "Command Data" on page 128.

Table 10. Parameters for Special Commands

Command Parameter	Description
AppletMode	For @CreatePopupFrame, specifies the mode for an applet to be displayed in the new browser window.
Dimension	For @CreatePopupFrame, specifies the dimension of the new browser window.
PageURL	For @CreatePopupFrame, specifies a URL to be displayed in the new browser window.
SWECommand	For @InvokeSWECommand, specifies a Siebel Web Engine (SWE) command to be invoked.
View	For @ViewWorkObject, specifies the name of a view.

Examples of Special Commands

This topic provides examples of the special commands.

Example of @Associate Special Command

Table 11 and Table 12 provide an example command using the @Associate special command. This example defines a command and associated command data that creates an association between the current activity object and the current account object. The example command is enabled only when the agent is working with accounts and when a work-item tracking object exists. For more information, see the description for @Associate in Table 9 on page 88.

Table 11. Command: AssociateAccount

Parameter Name	Parameter Value
DeviceCommand	@Associate
Hidden	True

Table 12 shows a command data definition for this command definition.

Table 12. Command Data: AssociateAccount

Parameter Name	Parameter Value
BusComp	Account
Param."Account Id"	{Id}

Example of @ViewWorkObject Special Command

Table 13 and Table 14 provide an example command using the @ViewWorkObject special command. This example defines a command and associated command data that allows a user to view the work-item tracking record. The command specifies the menu item to be used for viewing the record and assigns the command to the hot key SHIFT+F8. The command is enabled only when a work-item tracking object exists. For more information, see the description for @ViewWorkObject in Table 9 on page 88.

Table 13. Command: ViewWorkObject

Parameter Name	Parameter Value
DeviceCommand	@ViewWorkObject
HotKey	SHIFT+F8
MenuPosition	8
Title	View Work Item Object

Table 14 shows a command data definition for this command definition.

Table 14. Command Data: ViewWorkObject

Parameter Name	Parameter Value
Param.View	Activity Attachment View

Event Handlers

This topic describes event handlers. It contains the following information:

- "Event Handler Parameters" on page 95
- "Handling an Inbound Call Received by an Agent" on page 99

Communications events from the communications system are directly passed to Siebel Communications Server, in particular the Communications Session Manager, for further processing.

NOTE: Each event log specified in a single event response must be unique. (Event logs can be specified in the All Event Handlers view, but they are actually associated with the event response that is associated with the event handler.) For more information, see "Event Responses" on page 100.

Event Handler Process Overview

Communications events involve the following phases of activity:

1. An event or action, such as a phone call being disconnected, occurs in the communications system, such as a telephony switch. The switch forwards events to a communications middleware server, such as a CTI middleware server.
2. The middleware server, such as CTI middleware, forwards the event to the Communications Client business service.
3. The Communications Client business service processes the event and executes any actions defined in the configuration data in the database, or forwards events to business service methods or to Siebel VB or Siebel eScript code.

   By working with event data in the Administration - Communications screen, you can define the actions to be performed when a particular communications event is received. Such actions are invoked immediately upon receipt of such an event.

   Event handlers specify what kind of event from the communications system is processed and specify which event response is called as a result.

Event Handler Parameters

Table 15 describes the parameters available in event handlers in the communications configuration data. The value of the Macro column indicates whether macro expansion applies to the parameter.

Table 15. Event Handler Parameters

Parameter	Type	Macro	Description
DeviceEvent	Char	Not applicable	Communications device-generated event name. Device commands that are recognized by the communications driver are specific to particular communications systems. Possible values include the device events supported by your communications driver. For information about the events supported by the Virtual CTI driver, see "Virtual CTI Driver Events" on page 253. **NOTE:** Do not specify this parameter as an event handler parameter. Rather, specify the device event for the event handler directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Filter	Group	Not applicable	Event data-field filter. This works like a standard IsLike() function. All filter results are logically joined by AND to determine if the event matches this event handler. You can filter data received from the communications driver, data attached to the work item by the channel manager, or data attached or modified by an agent who previously handled the work item, such as a transferring agent. The Filter and FilterSpec parameters can be used together to evaluate event data-field data received with the device event associated with this event handler. Evaluations using the Filter parameter (AND conditions) occur before evaluations using the FilterSpec parameter (compound predicate). The fields available for filtering depend on the communications driver you are using.

Table 15. Event Handler Parameters

Parameter	Type	Macro	Description
FilterSpec	Char	Yes	Event data-field filter that supports simple or complex queries. Filter results are evaluated using a compound predicate that can include standard query operators to determine if the event handler matches the event. FilterSpec queries use standard comparison operators, including: = LIKE AND OR EXISTS > < >= <= For example: FilterSpec = "[*attr1*] IS NOT NULL OR [*attr2*] LIKE '*value**'" where *attr1* and *attr2* are event data fields and *value* represents part of a comparison value for *attr2*. In this example, the event handler is evaluated for a match if *attr1* is not empty or if *attr2* is like the value of *value**. Here is another example: FilterSpec = "[@UserName]='SADMIN'" In this example, @UserName is macro-expanded before the query is performed. The Filter and FilterSpec parameters can be used together to evaluate event data-field data received with the device event associated with this event handler. Evaluations using the Filter parameter (AND conditions) occur before evaluations using the FilterSpec parameter (compound predicate). For more information, see the descriptions for the Filter event handler parameter and the QuerySpec event response parameter.

Table 15. Event Handler Parameters

Parameter	Type	Macro	Description
Order	Numeric	Not applicable	The order in which event handlers (for which the same device event is specified) are tested against event matching. Each received communications event is checked against all event handlers to determine which response to execute. Event handlers with a lower Order value are checked before those with higher values. The default is 0. The check stops when there is a match. **NOTE:** Do not specify this parameter as an event handler parameter. Rather, specify the order for the event handler directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Profile	Char	Not applicable	Name of the communications driver profile that generates the device event associated with this event handler. If a profile is associated with the event handler, then the event handler is evaluated only for events received from the communications driver for that profile. For more information, see "Creating Event Handlers" on page 65. **NOTE:** Do not specify this parameter as an event handler parameter. Rather, specify the profile for the event handler directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Response	Char	Not applicable	Name of the event response that executes when a matching event is detected. **NOTE:** Do not specify this parameter as an event handler parameter. Rather, specify the event response for the event handler directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.

Table 15. Event Handler Parameters

Parameter	Type	Macro	Description
ServiceMethod	Char	Not applicable	The name of a Siebel business service and method to be called in order to evaluate the event handler for a match. You can use the ServiceMethod and ServiceParam parameters to supplement or replace filter mechanisms using the Filter or FilterSpec parameters. You can also use this method to execute custom code before executing a specified event response. The business method must set a parameter of Result to a value 1 or 0 (or True or False): ■ 1 (True) indicates that this event handler matches and is executed, and the associated event response is executed. ■ 0 (False) invalidates this event handler; the next event handler is evaluated. Specify the service and method in the form *service*.*method*. Optionally, you can use the ServiceParam parameter to provide parameter names and values to pass to the method to be called. For more information, see "Using Business Services with Siebel Communications Server" on page 181.
ServiceParam	Group	Yes	A group of subparameters that are parameters to the Siebel business service method (if any) invoked using the ServiceMethod parameter. You create each parameter you need in the form ServiceParam.*param_name*, then specify a parameter value. The parameter name and value are both passed to the service method. Order parameters in the sequence in which they are expected by the service method: ServiceParam.Param1 = "value1" ServiceParam.Param2 = "name" Param1 and Param2 are subparameters of the ServiceParam parameter. For more information, see "Using Business Services with Siebel Communications Server" on page 181.

Handling an Inbound Call Received by an Agent

Communications events provide the following general options for handling device events for calls received by an agent. Such an inbound call can employ events like call incoming or event ringing, or call connected or event established, as defined for a communications system employing CTI.

The following list describes the options for handling device events:

- **Handle when call connected after agent answers (default).** In this scenario, the agent answers the call, triggering the event handler.

  The phone rings. The agent clicks Accept Work Item. The call connected or event established type of event is received. This event triggers handling such as to perform a query to generate a screen pop. In this case, an agent's work is not interrupted by an unexpected screen pop.

- **Handle immediately, when phone rings.** In this scenario, the phone rings, triggering the event handler (before the agent answers the call).

  The phone rings. The call incoming or event ringing type of event is received. The event triggers handling such as to perform a query to generate a screen pop. The agent clicks Accept Work Item, then the connection is established. In this case, an agent's work might be interrupted by a screen pop when the phone rings, without any notifications or confirmations.

NOTE: If the user has modified the current record, then this record is automatically committed anytime an agent clicks a button on the communications toolbar or anytime a screen pop is triggered. If the user has not provided values for all required fields, then a pop-up window prompts the user to provide the values to complete the record.

The Siebel sample communications configurations provide examples of some of these scenarios. You can implement other screen-pop behavior using Siebel VB or Siebel eScript.

Example of Using Event Handler to Handle Inbound Call

Here is an example event handler, named InboundCallReceived, that is based on the default inbound call screen-pop scenario previously described in the scenario *Handle when call is connected after agent answers*. For this event handler:

- Set the Order field to 0 to make sure it takes priority over any other event handlers that might otherwise match.
- Set the Device Event field to *event_name*, where *event_name* corresponds to an event such as the agent clicking Accept Work Item.
- Include the following event handler parameter:

  Filter.ANI = "*"

The example event handler is associated with an event response OnInboundCallReceived. Because the Order field is set to 0, this event handler is checked first for any event received where the device event is as specified. The event handler filters the event data field ANI. If this field exists, then the event matches and invokes the event response OnInboundCallReceived.

To handle a call-incoming or event-ringing type of event without waiting for the agent to click Accept Work Item, you can create a similar event handler that triggers immediately when a suitable event occurs. You might name such an event handler ImmediateRingingHandler and invoke a similarly named response.

Event Responses

This topic describes event responses. It contains the following information:

- "Event Response Parameters" on page 102
- "Event Response Example" on page 113

Event responses in the communications configuration data specify how the Siebel client associates response actions with an event. Each event handler specifies a response that corresponds to an event response. Different event handlers can initiate the same event response.

NOTE: Each event log specified in a single event response must be unique. When you associate event logs with an event response in the Siebel client, the user interface prevents you from associating the same event log more than once. For example, if you have already associated the event log OutboundActivityContactCall, using type Log, then you cannot associate the same event log using another Log Type, such as SingleLog. If you create DEF files to import, then do not specify, within an event response definition, multiple event log parameters that specify the same value. If you do so, then an error is generated when you attempt to import this configuration data.

Event Response Process Overview

In executing the event response, Siebel Communications Server performs the following actions:

1. If a Siebel business service method name is specified, such as to update the customer dashboard, then that method is invoked.

   If the method does not return the required result, such as a value of True (1) for the output argument Continue, then the response stops executing. If an event log of type Log is specified, then that event log is processed. Go to Step 13 on page 102.

   The business service method can also alter the event data fields that were associated with the work item, such as to add new fields.

   For more information about the parameters mentioned here, see Table 16 on page 102.

2. If a Siebel VB or Siebel eScript script name is specified, then that script is invoked.

   If the script does not return the required result, such as Continue, then the response stops executing. If an event log of type Log is specified, then that event log is processed. Go to Step 13 on page 102.

   For more information about invoking a Siebel VB or Siebel eScript script from an event response, see "Integrating with Siebel Scripting Languages" on page 187, and see the descriptions for the Script and ScriptParam parameters in Table 16 on page 102.

3. If a communications command is specified, then the command is executed.

4 If the UseCtxData parameter is defined and set to True, then Communications Server checks to determine if the event has screen context data attached.

   If screen context data is available, then the corresponding screen and view are displayed. This behavior is applicable only for screen pops and screen transfers. (In addition, the Thread Applet and Thread Field properties must be set for the view in order to support screen pops and screen transfers.)

   If an event log of type ContextLog is specified, then that event log is processed. If ContextLog is not specified, but an event log of type Log is specified, then that event log is processed. Go to Step 13 on page 102.

5 If the QueryBusObj, QueryBusComp, and QuerySpec parameters are defined, then the query specified by QuerySpec is macro-expanded and executed. If these parameters are not defined, then go to Step 9; otherwise, go to Step 6.

   Step 7 and Step 8 define the views upon which this query is displayed.

6 If there is only one row in the query result from QuerySpec, and if the QueryBusComp2 and QuerySpec2 parameters are defined, then the query specified by QuerySpec2 is macro-expanded and executed. If these parameters are not defined, then go to Step 9; otherwise, go to Step 7.

7 If there is only one row in the query result from QuerySpec, then that row is displayed in the view specified by the SingleView parameter. Any field specified by SingleField is populated with data defined in the parameter value. Go to Step 9.

8 If there are several rows in the query result from QuerySpec, then those rows are displayed in the view specified by the MultiView parameter. Go to Step 9.

9 If the OpenView parameter is specified, then the view is displayed that is specified by OpenView. Go to Step 10.

10 If the AddBusComp, AddBusObj, AddRecordView, AddRecordApplet, and AddField parameters are defined, then the view is displayed that is specified by AddRecordView.

   A new record is created in the applet specified by AddRecordApplet. Any field specified by AddField is populated with data defined in the parameter value. Go to Step 13 on page 102.

11 If a Siebel SmartScript script name or other data is specified to be passed to Siebel SmartScript, then that SmartScript is invoked.

   For more information about invoking a SmartScript from an event response, see "Integrating with Siebel SmartScript" on page 193 and see the description for the SmartScript parameter in Table 16 on page 102.

12 If the FindDialog and FindField parameters are defined, then the Search Center appears, populated with search criteria. The Search Center does not appear if the OpenView parameter is also specified as in Step 9.

13 If an event log is associated with this event response, then a log record is created using the object defined in the event log, for example, an activity record.

Multiple event logs can be associated, one of which matches, depending on how the event response executed. For example:

- If UseCtxData is invoked, as in Step 4 on page 101, then a ContextLog log can be created.
- If SingleView is invoked, as in Step 7 on page 101, then a SingleLog log can be created.
- If MultiView is invoked, as in Step 8 on page 101, then a MultiLog log can be created.
- If AddBusComp, AddBusObj, AddRecordView, AddRecordApplet, and AddField are invoked, as in Step 10 on page 101, then an AddLog log can be created.
- If FindDialog is invoked, as in Step 12, then a FindLog log can be created.
- Otherwise, a log can be created as specified by Log.

For more information about event logs, see "Event Logs" on page 113.

Event Response Parameters

Table 16 describes the parameters available in event responses. The value of the Macro column indicates whether macro expansion applies to the parameter.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
AddBusComp	Char	Not applicable	Business component name to use when adding a new record.
AddBusObj	Char	Not applicable	Business object name to use when adding a new record.
AddField	Group	Yes	Field names and predefined value when adding a new record.
AddLog	Char	Not applicable	Event object to execute if the application navigates to the view and applet specified by the AddRecordView and AddRecordApplet parameters. AddLog determines the Siebel activity record to be created in response to communications events. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
AddRecordApplet	Char	Not applicable	Name of the applet that is displayed, in the view specified by AddRecordView, when adding a new record. If AddRecordApplet is not specified, then the record is created in the first applet that can be inserted which has the same business component as specified with AddBusComp.
AddRecordView	Char	Not applicable	Name of the view that is displayed when adding a new record.
Command	Char	Not applicable	Optionally specifies the name of a command in the communications configuration that is invoked when this event response is invoked.
ContextLog	Char	Not applicable	Event object to execute if screen-context data is used for screen display. Determines the Siebel activity record to be created in response to communications events. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
FindDialog	Char	Not applicable	Find Object name to determine what is displayed in the Search Center if no rows are returned.
FindField	Group	Yes	Find Field names and default values to determine what is displayed in the Search Center if no rows are returned.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
FindLog	Char	Not applicable	Event object to execute if the Search Center is displayed, as specified using FindDialog and FindField. Determines the Siebel activity record to be created in response to communications events. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Log	Char	Not applicable	Event log to use if no other log is specified for an action. Determines the Siebel activity record to be created in response to a communications event. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
MultiLog	Char	Not applicable	Event object to execute if the view specified using the MultiView parameter is displayed. Determines the Siebel activity record to be created in response to communications events. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
MultiView	Char	Not applicable	Name of the view that is displayed if the query result contains more than one row.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
OpenView	Char	Not applicable	Name of the view that is displayed, regardless of query results. The view specified by OpenView is displayed only if a view specified by SingleView or MultiView is not displayed.
QueryAfterAnswer	Boolean	Not applicable	A parameter that, when set to True, causes Communications Server to answer the call first and then execute the screen-pop query. When set to False (the default setting), this parameter causes Communications Server to execute the query first and then answer the call. This parameter is valid only for an event response invoked by an event handler that follows the *handle immediately* model. For more information about screen pops, see "Handling an Inbound Call Received by an Agent" on page 99.
QueryBusComp	Char	Not applicable	Business component name for the query specified by QuerySpec.
QueryBusComp2	Char	Not applicable	Business component name for a second query, specified by QuerySpec2. The second query executes only if this parameter is specified and if the first query (specified by QuerySpec) returns exactly one row. This feature can be used to get screen pops related to campaigns. For example, the first query can locate the campaign that matches the call type, using DNIS to identify the number the caller dialed. The second query can locate the contact (within that campaign) whose work number matches that of the caller, using ANI to identify a caller's number.
QueryBusObj	Char	Not applicable	Business object name containing the business components specified by QueryBusComp and QueryBusComp2.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
QueryFields	Multi	Not applicable	Field names resulting from the business component query specified by QuerySpec. Only the fields specified here are activated when the Siebel application performs the query. Since the same business component is used to navigate to the destination view, you can use this feature to specify the fields required for that view. If nothing is specified here, then the navigation process causes the business component to be queried again. QueryFields can speed up a screen-pop navigation by preventing additional queries of an entire business component.
QueryFields2	Multi	Not applicable	Field names resulting from the business component query specified by QuerySpec2. For more information, see the description of the QueryFields parameter.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
QuerySpec	Char	Yes	Standard business component query. The query uses business component field names from the business object and business component specified with the QueryBusObj and QueryBusComp parameters. (No other fields can be specified.) Simple and complex queries are supported, using standard comparison operators, including: = LIKE AND OR EXISTS > < >= <= For example, if QuerySpec is defined as follows, then the query matches records in the current business component (assuming the business component contains the fields indicated) if either the work phone number or the home phone number matches the ANI value for the call: 'Work Phone #'='{ANI}' OR 'Home Phone #'='{ANI}' Alternatively, if QuerySpec is defined as follows, then the query matches records in the current business component (example is for Action business component) if the current user is one of the owners specified for the multivalue field Owned By: EXISTS('Owned By'='{@UserName}') The keyword EXISTS *must* be used when defining a query specification for a multivalue field.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
QuerySpec2	Char	Yes	A second query that might be performed, after the query specified by the QuerySpec parameter. QuerySpec2 queries the business component specified by the QueryBusComp2 parameter. (Both QueryBusComp and QueryBusComp2 specify business components in the business object specified by the QueryBusObj parameter.) The result of the QuerySpec2 query can be used by Siebel SmartScript. For more information on how the query can be constructed, see the description for the QuerySpec parameter.
Script	Char	Not applicable	Siebel VB or Siebel eScript script name to invoke (if specified). Parameters are passed using the ScriptParam parameter. For more information about using Siebel VB or Siebel eScript, see "Integrating with Siebel Scripting Languages" on page 187.
ScriptParam	Group	Yes	A group of subparameters that are parameters to the script method (if any) invoked using the Script parameter. You create each parameter you need in the form ScriptParam.*param_name*, then specify a parameter value that is passed to the script. The parameter names themselves are not passed to the script. Order parameters in the sequence in which they are expected by the script: ScriptParam.Param1 = "value1" ScriptParam.Param2 = "name" Param1 and Param2 are subparameters of the ScriptParam parameter.
ServiceMethod	Char	Not applicable	The name of a Siebel business service and method to be called. The service and method are specified in the form *service.method*. Optionally, you can use the ServiceParam parameter to provide parameter names and values to pass to the method to be called. For more information, see "Using Business Services with Siebel Communications Server" on page 181.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
ServiceParam	Group	Yes	A group of subparameters that are parameters to the Siebel business service method (if any) invoked using the ServiceMethod parameter. You create each parameter you need in the form ServiceParam.*param_name*, then specify a parameter value. The parameter name and value are both passed to the service method. Order parameters in the sequence in which they are expected by the service method: ServiceParam.Param1 = "value1" ServiceParam.Param2 = "name" Param1 and Param2 are subparameters of the ServiceParam parameter. For more information, see "Using Business Services with Siebel Communications Server" on page 181.

SingleField	Group	Yes	When the result from the QuerySpec query is exactly one record, that record can be updated if you provide a field name and predefined field value, using this parameter.
			
			For example, in the following event response, QuerySpec likely returns a single record, then SingleField can update the Primary Owned By field to the current agent:
			
			[EventResponse:EmailWorkStarted]
			QueryBusObj = "Email Response"
			QueryBusComp = "Action"
			QuerySpec = "Id = '{ActivityID}'"
			SingleView = "Communication Detail - Response View"
			SingleField.'Primary Owned By' = "{@UserName}"
			Log=EmailWorkStarted

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
SingleLog	Char	Not applicable	Event object to execute if the view specified using the SingleView parameter is displayed. Determines the Siebel activity record to be created in response to communications events. For an example of using this parameter, see "Event Response Process Overview" on page 100. **NOTE:** Do not specify this parameter as an event response parameter. Rather, specify an event log of this type for the event response directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
SingleView	Char	Not applicable	Name of the view that is displayed if the result from the QuerySpec query is exactly one row. QueryBusComp2, QuerySpec2, and QueryFields2 can be used to query a second business component of the QueryBusObj for a single record.
SmartScript	Group	Yes	Parameter to be passed to Siebel SmartScript. Can include a script name or other data to pass to SmartScript. You can specify any of the following subparameters: SmartScript.CampaignId SmartScript.CampContactId SmartScript.ContactId SmartScript.LanguageCode SmartScript.ScriptId SmartScript.ScriptName Either SmartScript.ScriptName or SmartScript.ScriptId must be specified in order to launch a particular SmartScript. For more information, see "Integrating with Siebel SmartScript" on page 193.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
UseCtxData	Boolean	Not applicable	Enables or disables automatic use of screen-context data attached to an event. When UseCtxData is True, and the event contains a Siebel screen bookmark, the bookmark is used to perform a screen pop or screen transfer. The default is False. **NOTE:** The size of a Siebel screen bookmark is defined by the middleware and communications driver and might vary according to each vendor.

Table 16. Event Response Parameters

Parameter	Type	Macro	Description
WorkObject	Char	Yes	Specifies data identifying a work-item tracking database record that can be written to the database. This parameter can make work-item tracking possible even if your communications system does not fully support attaching data to a work item. The WorkObject parameter is not necessary for CTI middleware packages that support call data attachment. This data is passed to the communications driver, which attaches the data to the call without requiring the WorkObject parameter. If you are using a custom driver developed for other middleware, then you might have to use the WorkObject parameter in order to pass work-item tracking data from one agent to another, such as when a call is transferred or made as a conference call. Example: An agent receives a call and creates the activity record for it. When this agent transfers the call to a second agent, the second agent wants access to the same activity record. For the first agent's session, the row ID of the activity record is encoded, and the call is transferred. If the WorkObject parameter is defined appropriately, then the second agent can continue tracking the call-activity record. Format values for this parameter as follows: *bus_obj_name*;*bus_comp_name*;*row_ID* For activity records, Action is the name of the business object and the name of the business component. Whether you require the WorkObject parameter, and how you define the parameter, depend on your communications system and driver. For example, with Aspect middleware, if variable E is used to store the row ID, then define WorkObject as follows: Action;Action;{E} Whether this parameter definition works as intended depends on whether the communications driver expects to receive data in this form to pass to the receiving agent.

Event Response Example

Table 17 shows an example of an event response.

Table 17. Event Response: OnInboundCallReceived

Parameter Name	Parameter Value
QueryBusObj	Contact
QueryBusComp	Contact
QuerySpec	'Work Phone #'='{ANI}'
SingleView	Service Contact Detail View
MultiView	Contact List View
FindDialog	Service Request
FindField.CSN	Ask Caller
SingleLog	LogIncomingCallContactFound
MultiLog	LogIncomingCallMultiContactFound
FindLog	LogIncomingCallContactNotFound

In this example, the information specified with the business object and business component parameters is retrieved. Communications Server then performs a query on the phone number retrieved from the ANI event data field, to see if it matches a value in the Work Phone # field in the Contact business component. (SingleLog, MultiLog, and Log are specified using fields, rather than parameters.)

If QuerySpec returns a single record, then the agent is navigated to the service requests view specified by SingleView. The event log LogIncomingCallContactFound, which is specified by the SingleLog parameter, creates a call activity record automatically.

If QuerySpec returns multiple records, then the agent is navigated to the contacts view specified by MultiView. The event log LogIncomingCallMultiContactFound, which is specified by the MultiLog parameter, creates a call activity record automatically.

If QuerySpec returns no records, then the Search Center appears and displays *Ask Caller* in the CSN field. The event log LogIncomingCallContactNotFound, which is specified by the FindLog parameter, creates a call activity record automatically.

Event Logs

This topic describes event logs. It contains the following subtopic: "Event Log Parameters" on page 114.

Event logs in the communications configuration data define log-generation rules for communications activity. Some event responses have an associated event log.

NOTE: Each event log specified in a single event response must be unique. Define separate event logs for each such usage context, accordingly. For more information, see "Event Responses" on page 100.

Event Log Parameters

Table 18 describes the parameters available in event logs. The value of the Macro column indicates whether macro expansion applies to the parameter.

Table 18. Event Log Parameters

Parameter	Type	Macro	Description
AfterWork	Group	Yes	Field names and values, from the log business component, that are used to update the work-tracking log record when the work item has ended or been released. **NOTE:** After the work item has been released, the log record that is modified according to the fields specified with this parameter is refreshed when the agent clicks off the record, then clicks on it again.
BusComp	Char	Not applicable	Log business component name.
BusObj	Char	Not applicable	Log business object name.
Display	Boolean	Not applicable	Parameter that, when it is set to True and the current view contains a business component specified in the event log, uses the current business component to add a new log record. The log record appears on the screen immediately and is selected. In order for this parameter to function as intended, the current view must retain the focus. If Display is set to False, then the log record is generated but is not selected, and might not be displayed in the current record set. The default is False. **NOTE:** While the work item is active, the agent can modify the log record. After the work item has been released, however, if AfterWork is set, then an agent's changes might conflict with how the log record was written by the Siebel CRM system software at that time. The agent can click off the record, then click on it again to modify it.

Table 18. Event Log Parameters

Parameter	Type	Macro	Description
LogField	Group	Yes	Field names and values of the log business component.
QuerySpec	Char	Yes	A query specification for retrieving a record in the business component: ■ If QuerySpec is not defined, then a new record is created for the event log. ■ If QuerySpec is defined and returns a single record, then this record is used for the event log. ■ If QuerySpec is defined and returns more than one record, then a new record is created for the event log.
ServiceMethod	Char	Not applicable	The name of a Siebel business service and method to be called. The service and method are specified in the form *service*.*method*. Optionally, you can use the ServiceParam parameter to provide parameter names and values to pass to the method to be called. If ServiceMethod and ServiceParam are specified, then they are executed first when the event log is invoked. If the specified business service method does not return the required result, such as a value of True (1) for the output argument Continue, then the event log stops executing. For more information, see "Using Business Services with Siebel Communications Server" on page 181.
ServiceParam	Group	Yes	A group of subparameters that are parameters to the Siebel business service method (if any) invoked using the ServiceMethod parameter. You create each parameter you need in the form ServiceParam.*param_name*, then specify a parameter value. The parameter name and value are both passed to the service method. Order parameters in the sequence in which they are expected by the service method: ServiceParam.Param1 = "value1" ServiceParam.Param2 = "name" Param1 and Param2 are subparameters of the ServiceParam parameter. For more information, see "Using Business Services with Siebel Communications Server" on page 181.

Table 18. Event Log Parameters

Parameter	Type	Macro	Description
WorkTrackingObj	Group	Yes	Field name and values of a work-tracking object to create. You can retrieve the value of the tracking object from a command. For example, the following parameter defines a custom work-tracking object, called *ContactId*, whose value is derived from the Id field of the Contact business component. WorkTrackingObj.ContactId = "{Contact.Id}" For a selected work item, this data might then be accessed from a communications command, using the WorkTrackingObj command data parameter. In this example, ContactId becomes a custom work item attribute. Alternatively, if an event log includes a parameter definition like the following, then text you specify is displayed in the Work Items list in the communications toolbar. WorkTrackingObj.Description = "your descriptive text goes here" For more information about using these and other attributes, see "Work Item Attributes" on page 160.

Commands

This topic describes commands. It contains the following information:

- "Hierarchical Commands (Commands and Subcommands)" on page 117
- "Command Parameters" on page 119

Commands in the communications configuration data define the appearance and availability of communications commands.

By using the DeviceCommand parameter in a command, you can invoke a command on the communications system or invoke a special command.

- Device commands that are executed by the communications driver. Such commands are specific to particular middleware vendors, or to how Oracle supports communications toolbar interactivity. For information about the commands supported by the Virtual CTI driver, see "Virtual CTI Driver Commands" on page 244.
- Device commands that are independent of communications drivers. For more information, see "Special Commands for Device Commands" on page 87.

Communications commands might instead invoke a business service method, a SmartScript, or a Siebel VB or Siebel eScript script. For commands, each of these mechanisms is mutually exclusive: invoke only one of these mechanisms in a given command. (Scripts and business service methods take precedence over device commands.) For more information:

- About invoking a business service method, see "Using Business Services with Siebel Communications Server" on page 181 and Appendix B, "Siebel Communications Server Business Services."
- About invoking a Siebel VB or Siebel eScript script, see "Integrating with Siebel Scripting Languages" on page 187.

Hierarchical Commands (Commands and Subcommands)

Communications commands can be defined in hierarchical fashion; you can define group commands that contain subcommands. These subcommands are invoked when the group command is invoked. Grouping or nesting commands using subcommands helps to define the communications user interface in the Siebel application.

For more information about specifying subcommands, see "Defining Communications Commands" on page 67. For more information about how command groups are used in configuring the communications toolbar and menus, see Chapter 6, "Configuring User Interface Elements."

When you specify subcommands within group commands, note the following:

- Commands can have subcommands. A command containing subcommands is sometimes referred to as a group command.
- Do not define invalid recursive subcommand relationships, such as making a subcommand include its parent group command as a subcommand.
- The Subcommands list in the All Commands view specifies the order in which commands are executed. Alternatively, if the command parameter ExecuteAll is set to True, then all subcommands can execute. As with any command, subcommands can execute only when they are enabled. Subcommands that are disabled are skipped and not executed.
- If a subcommand fails when executing, then the remaining subcommands are not executed. However, if the command parameter ExecUntilOK is set to True, then the remaining subcommands are executed until one of them succeeds.
- Subcommands and their order are represented in DEF files you export by using the command parameter SubCommand_*N*, where *N* is the order value for each subcommand.
- At run time, commands that invoke device commands are enabled or disabled based in part on the status of the device command. For more information, see "SCCommandFlag" on page 268 and "CacheCommandInformation" on page 273.
- At run time, commands that invoke business service methods rather than device commands are determined to be enabled or disabled by invoking from the applicable business service the method CanInvokeMethod("*method_name*"), where *method_name* is your business service method to be invoked. If True is returned, then the method can be invoked. If False is returned, then the method cannot be invoked and the command is disabled.

- Several command parameters are used to determine whether a group command or subcommand matches and can be executed. Such command parameters include AllViews, CmdChannelOnFocus, FilterSpec, OnEditControl, and View.
- Several command data parameters are also used to determine whether a group command or subcommand matches and can be executed. Such command data parameters include BusComp, BusObj, Filter, and RequiredField.
- Commands specifying subcommands cannot also specify a device command or business service method.
- If you require a command that executes both a device command and a business service method, then define a group command where ExecuteAll is True and where two subcommands are specified. Define one subcommand to execute a device command, such as a make-call command, and define the other subcommand to invoke your business service method. If you need command data parameters, then create command data definitions and associate them with the subcommands.
- In some cases, a group command must not have command data associated with it. If a group command is at the top-level (is not itself a subcommand), or if a group command is also a subcommand and ExecuteAll is set to True for its parent group command, then command data might be associated. However, in the latter case, if ExecuteAll is not set to True for the parent group command, then do not associate command data with the child group command.
- The command parameter SelectApplet (and related parameters such as SelectBusObj, SelectBusComp, SelectView, and so on) can be used to display a dialog box that pertains to all subcommands in a group command, where ExecuteAll is True. To configure this, you must define these parameters in the top-level command and not in the individual subcommands. The dialog box applies to all of the subcommands.

Command Parameters

Table 19 on page 119 describes the available command parameters. The value of the Macro column indicates whether macro expansion applies to the parameter. For more information about configuring the communications toolbar and Communications submenu commands, see Chapter 6, “Configuring User Interface Elements.”

Table 19. Command Parameters

Parameter	Type	Macro	Description
AcceptReject	Boolean	Not applicable	Indicates whether the command is for prompting the agent to accept or reject a work item. Set to True for such a command. Execution of the device command is subject to the Accept or Reject dialog box and subject to the agent’s choices. If AcceptReject is True, then this dialog box displays, with the title specified by the ARTitle parameter. The work item matches the definition of the ARWorkItemID parameter: ■ If the agent accepts the work item, then the device command for this command can be executed. ■ If the agent rejects the work item, then the command specified by the AROnRejectCmd parameter is executed.
AllViews	Boolean	Not applicable	Parameter that, when set to False, disables the command on all views other than those specified using the View parameter. The default is True.
AROnRejectCmd	Char	Not applicable	Specifies the name of a command to be executed if the agent chooses to reject the work item. For example: AROnRejectCmd = "OnEmailWorkItemRejected"
ARTitle	Char	Not applicable	Provides text to be presented to the agent when the agent is prompted to accept or reject the work item. For example, for a command used for the accept or reject case for email work items, the parameter might be defined as follows: ARTitle = "Would You Like to Accept This Email Work Item?"

Table 19. Command Parameters

Parameter	Type	Macro	Description
ARWorkItemID	Char	Yes	Defines the ID of the work item to be accepted or rejected. For example, the parameter might be defined as follows: ARWorkItemID = "{$InboundWorkItem(Email, WorkItemID)}"
CmdChannel OnFocus	Boolean	Not applicable	Specifies whether the command can only be enabled when the work item with the focus has the same channel as the device command for this command. In other words, if CmdChannelOnFocus is True, then the current work item is of a particular channel, and the device command is valid for a work item of this channel, then this command can be enabled. Otherwise, the command is disabled. For example, the command is disabled if all of the following are true: ■ This parameter is set to True. ■ The device command for this command is *HoldCall* (command is for Voice channel). ■ The work item that currently has the focus (that is, the current item displayed in the Work Items list in the communications toolbar) is for a channel other than Voice. An equivalent way to filter the command is to define the FilterSpec parameter as follows: FilterSpec = [@SelectedWorkItem:ChannelType]='Voice'"

Table 19. Command Parameters

Parameter	Type	Macro	Description
CmdData	Char	Not applicable	The command data definition name that describes parameter generation for this command to be passed to a device command (or to a business service method, for example). This parameter can be empty if the device command does not require any parameters from a command data parameters, or if no device command is specified. For more information, see "Command Data" on page 128. **NOTE:** Do not specify this parameter as a command parameter. Rather, specify a command data definition for the command directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Description	Char	Yes	Description string for the command. This string appears in the ToolTip area when the user points to a communications toolbar button for this command. For more information, see "About Communications Toolbar Configuration" on page 135 and "Configuring Communications Menu Commands" on page 145.
DeviceCommand	Char	Not applicable	The device command executed when this communications command is executed. Device commands that are executed by the communications driver are specific to particular communications systems. Possible values include the device commands supported by your communications driver. For information about the commands supported by the Virtual CTI driver, see "Virtual CTI Driver Commands" on page 244. Device commands based on special commands are not executed by a communications driver. For more information, see "Special Commands for Device Commands" on page 87.

Table 19. Command Parameters

Parameter	Type	Macro	Description
ExecuteAll	Boolean	Not applicable	Default value is False. If you set ExecuteAll to True, then do not set ExecUntilOK to True. When used within a group command (containing subcommands), specifies that all subcommands can be executed. For example, if automatic login is in effect, and a default login command such as SignOnGroup is specified using the configuration parameter AutoLoginCmd, then all of this group command's subcommands execute when the agent starts the Siebel session. In this example, the agent is logged in to all applicable communications systems, such as ACD queues. For more information, see "Hierarchical Commands (Commands and Subcommands)" on page 117 and "Configuring Communications Login and Logout" on page 172.
ExecUntilOK	Boolean	Not applicable	Default value is False. If you set ExecUntilOK to True, then do not set ExecuteAll to True. When used within a group command (containing subcommands), specifies that each subcommand executes in sequence until one of them succeeds. If a subcommand fails to execute, then the next subcommand executes. For more information, see "Hierarchical Commands (Commands and Subcommands)" on page 117.

Table 19. Command Parameters

Parameter	Type	Macro	Description
FilterSpec	Char	Yes	Filter that supports simple or complex queries. Filter results are evaluated using a compound predicate that can include standard query operators to determine if the command matches. FilterSpec queries use standard comparison operators, including: = LIKE AND OR EXISTS > < >= <= Alternatively, FilterSpec can operate on the work item represented by the macro @SelectedWorkItem. For example, the following parameter definition matches voice work items: FilterSpec = "[@SelectedWorkItem:ChannelType]='Voice'" For more information, see the descriptions for the FilterSpec event handler parameter and the QuerySpec event response parameter.
Hidden	Boolean	Not applicable	Parameter that, when set to True, hides the command from the Communications submenu. This command is typically set to True for commands to appear only in the communications toolbar, or for commands that do not appear in the user interface. Set this parameter to False (the default) or omit the parameter in order for the command to appear in the Communications submenu. Use the Description, HotKey, MenuPosition, and Title parameters to specify additional settings relevant to the command's appearance in the Communications submenu. For more information, see "About Communications Toolbar Configuration" on page 135 and "Configuring Communications Menu Commands" on page 145.

Table 19. Command Parameters

Parameter	Type	Macro	Description
HotKey	Char	Not applicable	Keyboard shortcut to assign to this command, for use in the Communications submenu. This can be a combination of CTRL, ALT, or SHIFT and one of the keyboard letter or function keys, or just a single letter or function key. For example, valid values for the HotKey parameter include F12, CTRL+SHIFT+F, and so on. Keyboard shortcuts defined for communications commands do not use as input any data entered into the text entry field in the communications toolbar. **NOTE:** If you specify keyboard shortcuts, using the HotKey parameter, that conflict with keyboard shortcuts (accelerators) specified for commands in Siebel Tools, then the shortcuts for Siebel Tools take precedence over shortcuts assigned to this command by using the HotKey parameter. For more information, see "Configuring Communications Menu Commands" on page 145.
HotKeyText	Char	Not applicable	Specifies locale-dependent text to represent the keyboard accelerator for the command's menu item. The keyboard accelerator itself is specified using the HotKey parameter. This locale-dependent text appears in the Communications submenu. If HotKeyText is not used, then the text specified in HotKey is used instead. For more information, see "Configuring Communications Menu Commands" on page 145.
IndicateActiveCmd	Boolean	Not applicable	Parameter that, when set to True, specifies whether the toolbar button for this command must display the correct channel-specific icon for the device command for the active subcommand. This parameter is used in group commands containing subcommands for multiple channels, such as the AcceptWorkGroup command, for the Accept Work Item toolbar button. The default is False.

Table 19. Command Parameters

Parameter	Type	Macro	Description
LocalMenu	Boolean	Not applicable	Parameter that, when set to True, enables this command so it appears in the context-sensitive applet-level menu. This feature lets you, for example, define an applet-level menu item called Call Contact that is available when an agent works with contact records. The default is False. For more information, see "Configuring Communications Menu Commands" on page 145.
MenuPosition	Char	Not applicable	Parameter setting that determines the order in which the command appears in the Communications submenu or in the applet-level menu. Commands appear in ascending order with respect to the MenuPosition values: those with lower MenuPosition values appear above those with higher MenuPosition values. Values for the MenuPosition parameter can also specify additional menu command levels, within the Communications submenu of the Tools application-level menu. The applet-level menu commands, however, are all on a single level. For example, assume the SignOnGroupInMenu command (with Title parameter value *Log In*) has a MenuPosition value of 20. This group command appears in the Communications submenu, below any items with lower number, and in turn has subcommands representing submenu items. The subcommand LoginToPBX has values of 20.1, 20.2, and 20.3, specifying that it appears as a submenu item to SignOnGroupInMenu. LoginToPBX, with Title parameter value *Log In (Phone)*, would be available to agents at the following path in the application-level menu: Tools, then Communications, then Log In, then Log In (Phone) For more information, see "Configuring Communications Menu Commands" on page 145.

Table 19. Command Parameters

Parameter	Type	Macro	Description
MultiActiveCmdIcon	Char	Not applicable	Specifies the name of the icon file to use for the toolbar button, such as the Accept Work Item button, if multiple work items have arrived, such as a phone call and an email message. For example, the parameter can be defined as follows: MultiActiveCmdIcon = "misc_work.gif" For more information about toolbar button configuration, see "About Communications Toolbar Configuration" on page 135. For more information about icon files associated with communications drivers, see "Icon Files for Interactive Communications Drivers" on page 37.
OnEditControl	Boolean	Not applicable	Parameter that, when set to True, means that the command requires that the edit field in the communications toolbar has the focus and contains data. The default is False.
Profile	Char	Not applicable	Name of the communications driver profile that supports the device command associated with this command. If a profile is associated with the command, then the command can only be sent to the communications driver for that profile. For more information, see "Creating Commands" on page 69. **NOTE:** Do not specify this parameter as a command parameter. Rather, specify the profile for the command directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Script	Char	Not applicable	Siebel VB or Siebel eScript script name to invoke (if specified). Parameters are passed using the ScriptParam parameter for the associated command data definition. For more information about using Siebel VB or Siebel eScript, see "Integrating with Siebel Scripting Languages" on page 187.

Table 19. Command Parameters

Parameter	Type	Macro	Description
ServiceMethod	Char	Not applicable	The name of a Siebel business service and method to be called. The service and method are specified in the form *service.method*. Optionally, you can use the ServiceParam parameter for the associated command data definition to provide parameter names and values to pass to the method to be called. For more information, see "Using Business Services with Siebel Communications Server" on page 181.
SubCommand_*N*	Multi	Not applicable	The name of a command that is specified as a subcommand for the current command. The commands might appear to an agent as one command because only one of them is active at any point, depending on the Siebel context and the specified Order values for the subcommands. Alternatively, all subcommands can execute, if the ExecuteAll parameter is set to True. In a DEF file (the only place this parameter appears), the suffix *N* represents the order value for the subcommand. For example, a call-transfer command might be either to transfer to an employee or to transfer to a service request owner, depending on the current business component. The commands that are invoked from the communications toolbar and the Communications submenu often invoke other commands, which are specified as subcommands. **NOTE:** Do not specify this parameter as a command parameter. Rather, specify subcommands for the command directly, using the Administration - Communications screen. This parameter is used in DEF files that you export to or import from.
Title	Char	Not applicable	Name of the Communications submenu item associated with this command. If no Title parameter is specified, then the name of the device command serves as the menu item name. For more information, see "Configuring Communications Menu Commands" on page 145.

Table 19. Command Parameters

Parameter	Type	Macro	Description
View	Multi	Not applicable	Names of the views where this command is enabled if the AllViews parameter value is False.

Command Data

This topic describes command data.

Command data definitions in the communications configuration data define how communications command parameters are generated for use with a corresponding command. For example, a command data definition can instruct one of the transfer commands to transfer the caller to the current service request owner or can display a specified Siebel view and extract the transfer destination from that view.

Command Data Parameters

Table 20 describes the parameters available in command data definitions. The value of the Macro column indicates whether macro expansion applies to the parameter.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description
AttachContext	Boolean	Not applicable	Parameter that, when set to True, allows automatic passing of current screen-context data with the current work item. This is the easiest way of transferring screens between agents. AttachContext applies to voice work items only. The default is False. The Send Screen Pop and Receive Screen Pop settings in the Communications options of the User Preferences screen override the setting of the AttachContext command parameter. For more information, see "Setting Communications User Preferences" on page 217. **NOTE:** The size of a Siebel screen bookmark is defined by the middleware and communications driver and might vary according to each vendor.
BusComp	Char	Not applicable	Business component name. If this parameter is empty, then the associated command is enabled on all views. If a business component name is specified, then the associated command is enabled only on views that are based on the specified business component.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description
BusObj	Char	Not applicable	Business object name. If this parameter is empty, then the associated command is enabled on all views. If a business object name is specified, then the associated command is enabled only on views that are based on the specified business object.
OnField	Char	Not applicable	Field name that is required to be active. If this property is specified, then the related command is enabled only when this field is active (that is, when a cursor is in this field).
Param	Group	Yes	Parameter names and values for the command. You can use this parameter to pass data to accompany the device command, whether for a communications driver or a special command. You create each parameter you need in the form Param.*param_name*, then specify a value for the parameter that you pass for the device command. Parameters are specific to commands for particular middleware vendors, and possible values for these parameters depend on the communications driver. Example syntax follows for a subparameter *CmdParamName* and an event field named *EventFieldName*. This parameter value can be used to fetch event data defined as a child property set. Param.*CmdParamName* = {EventFields.*EventFieldName*} The Param parameter can also be used to pass parameter values received from outside Communications Server, such as from a business service that invokes a Siebel communications command. For more information, see "About Using Business Services with Events and Commands" on page 183.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description
RequiredField	Group	Yes	Field name and filter pair that are used as a criterion to determine if this command is active for this condition. RequiredField is always based on the currently selected row. For example, the following property causes the specified command to be disabled if field A of the current (selected) row in the current (active) applet does not contain a value that ends with X2. RequiredField.A = "*X2"
ScriptParam	Group	Yes	A group of subparameters that are parameters to the script method (if any) invoked using the Script parameter. You create each parameter you need in the form ScriptParam.*param_name*, then specify a parameter value that is passed to the script. The parameter names themselves are not passed to the script. Order parameters in the sequence in which they are expected by the script: ScriptParam.Param1 = "value1" ScriptParam.Param2 = "name" Param1 and Param2 are subparameters of the ScriptParam parameter.
SelectApplet	Char	Not applicable	An applet from which an agent can make a selection, such as to select a recipient for a work item or to specify a reason code. You can specify the following applets: ■ Accept Reject Popup Applet ■ ACD Transfer Call Applet ■ Transfer Multiple LOV Popup Applet
SelectBusComp	Char	Not applicable	Business component for an applet from which an agent can make a selection.
SelectBusObj	Char	Not applicable	Business object specifying application elements from which an agent can make a selection.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description
SelectParam	Boolean	Not applicable	Parameter that, when set to True, enables an applet in which the user can make a selection. For example, the applet can be used to select the person to whom to send, transfer, or conference a work item (such as to call or to transfer a call to) or used to specify a reason code. For example, for a MakeCall command, a dialog box can be displayed from which the agent can select an employee to call. The SelectBusObj, SelectBusComp, SelectApplet, SelectTitle, and SelectQuerySpec parameters determine the specific nature of the selection applet. The default is False.

SelectQuerySpec	Char	Yes	A query specification for an applet, such as Transfer Multiple LOV Popup Applet, from which an agent can make a selection.
			
			For example, a command data definition can use SelectQuerySpec as follows:
			
			[CmdData:NotReadyWithPopup]
			SelectParam = "True"
			SelectTitle = "Select the reason for changing status to Not Ready"
			SelectApplet = "Transfer Multiple LOV Popup Applet"
			SelectBusObj = "List Of Values"
			SelectBusComp = "List Of Values"
			SelectQuerySpec = "[Type]='REASON_CODE' AND [Active] = 'Y'"
			Param.Reason = "[Value]"
			
			In this example, SelectQuerySpec queries the List of Values business component to obtain the values to display in the Reason Codes list.
			
			If the applet allows you to select multiple records, as does Transfer Multiple LOV Popup Applet, then the selection values are concatenated as comma-separated values.

SelectTitle	Char	Not applicable	The title for the applet (dialog box) from which an agent can select a recipient for a work item. If this value is not specified, then the applet has the title that is assigned to the selection applet.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description
ServiceParam	Group	Yes	A group of subparameters that are parameters to the Siebel business service method (if any) invoked using the ServiceMethod parameter for the associated command. You create each parameter you need in the form ServiceParam.*param_name*, then specify a parameter value. The parameter name and value are both passed to the service method. Order parameters in the sequence in which they are expected by the service method: ServiceParam.Param1 = "value1" ServiceParam.Param2 = "name" Param1 and Param2 are subparameters of the ServiceParam parameter. For more information, see "Using Business Services with Siebel Communications Server" on page 181.

Table 20. Command Data Parameters

Parameter	Type	Macro	Description

WorkTrackingObj	Group	Yes	Field name and values of a work-tracking object previously specified in an event log definition. The WorkTrackingObj command data parameter stores or updates this object for the currently selected work item.
			
			For example, the following command and command data definitions update the customer dashboard with a contact ID:
			
			[Command:UpdateDashboardFromContact]
			Hidden = "True"
			ServiceMethod = "Persistent Customer Dashboard.Update Dashboard from CTI"
			CmdData = "UpdateDashboardFromContact"
			
			[CmdData:UpdateDashboardFromContact]
			BusComp = "Contact"
			ServiceParam.Field = "Id"
			ServiceParam.Value = "{Id}"
			WorkTrackingObj.ContactId = "{Id}"
			
			The WorkTrackingObj parameter updates the attribute value of "ContactId" for the currently selected work item. In this example, ContactId is a custom work item attribute.
			
			You can also use WorkTrackingObj to update the Siebel bookmark. For example, using the following line in a command data definition for a command to resume a suspended work item (such as to remove a call from hold) prevents a screen pop from occurring when an agent resumes a work item. If this parameter is not defined as shown, then a screen pop might display the view the agent was on when the work item was suspended.
			
			WorkTrackingObj.ViewBookmark = ""
			
			For more information, see "Work Item Attributes" on page 160.

6 Configuring User Interface Elements

This chapter provides information about configuring user interface elements for communications features. It includes the following topics:

About Communications Toolbar Configuration

This topic explains how the communications toolbar buttons relate to commands defined for the communications configuration and provides information about modifying the communications toolbar for your agents.

This topic contains the following information:

- "Communications Toolbar Items, Commands, and Methods" on page 136
- "Communications Toolbar Commands and Bitmaps" on page 137

For information about enabling communications activities through the communications toolbar, see "Enabling Session Communications" on page 201.

For information about using the communications toolbar, see "Using the Communications Toolbar" on page 223.

The communications toolbar is defined and configured, in part, within Siebel Tools, like other toolbars in the Siebel applications. Within Siebel Tools, the Communication toolbar object definition contains, or is linked to, a series of other object definitions, including toolbar items, commands, and bitmaps.

These object definitions, combined with associated business service methods, communications commands, and bitmap image files, determine the functioning and appearance of the communications toolbar for your agents.

NOTE: The available communications toolbar functions are also determined by functionality supported by applicable communications drivers and external communications systems, such as CTI middleware.

The communications toolbar connects to the Web server using HTTP. Toolbar functions are determined by communications events and commands, which interact with communications drivers loaded by the Communications Session Manager component. Communications between the driver and applicable middleware is subject to the implementation of the driver. For the high-interactivity client, the communications toolbar is a Java applet.

For more information about configuring object definitions for toolbars, toolbar items, commands, and bitmaps, see the developer documentation on the *Siebel Bookshelf*.

Related Books

Configuring Siebel Business Applications

Configuring Siebel Open UI

Using Siebel Tools

Siebel Developer's Reference

Communications Toolbar Items, Commands, and Methods

Table 21 displays information about elements associated with the communications toolbar items. From left to right, the table displays each toolbar item name, the associated Siebel Tools command name, and the business service method associated with the command (where applicable). The methods are those of the Communications Client business service. The Default column indicates whether the toolbar button appears in the toolbar by default. An asterisk (*) indicates that the button is part of a submenu. Each business service method name also corresponds to the name of a communications command. Communications configurations provided with Siebel Business Applications include commands with these names. For more information about these commands, see "Commands" on page 116.

Table 21. Communications Toolbar Items, Commands, and Methods

Toolbar Item	Tools Command (Display Name)	Business Service Method Or Communications Command	Default
Accept Email	Accept Incoming Email	AcceptEmailGroup	Yes*
Accept Misc Work	Accept Miscellaneous Work Item	AcceptMiscWorkGroup	No*
Accept Phone	Answer Phone Call	AnswerCallGroup	Yes*
Accept Work	Accept Work Item	AcceptWorkGroup	Yes
Blind Transfer	Blind Transfer Work Item	BlindTransferGroup	Yes
Consultive Conference	Consultative Conference Work Item	ConferenceTransferGroup	Yes

Table 21. Communications Toolbar Items, Commands, and Methods

Toolbar Item	Tools Command (Display Name)	Business Service Method Or Communications Command	Default
Consultative Transfer	Consultative Transfer Work Item	ConsultativeTransferGroup	Yes
Forward Work	Forward Work Item	ForwardWorkGroup	Yes
Hold Work	Hold Work Item	SuspendWorkGroup	Yes
Initiate Email	Initiate Email	SendEmailGroup	Yes*
Initiate Fax	Initiate Fax	SendFaxGroup	Yes*
Initiate Phone	Initiate Phone Call	MakeCallGroup	Yes*
Initiate SMS	Initiate SMS	SendSMSGroup	Yes*
Initiate Work	Initiate Work Item	InitiateWorkGroup	Yes
InQueue Time	InQueue Time	Not applicable	Yes
Media Indicator	Media Indicator	Not applicable	Yes
Not Ready	Not Ready State	NotReadyGroup	Yes
Not Ready Email	Not Ready for Email	NotReadyForEmailGroup	Yes*
Not Ready Phone	Not Ready for Phone	NotReadyForPhoneGroup	Yes*
Release Work	Release Work Item	ReleaseWorkGroup	Yes
Resume Work	Resume Work Item	ResumeWorkGroup	Yes
Retrieve Work	Retrieve Work Item	RetrieveWorkGroup	Yes
SignOff	Communication Sign Off	SignOffGroup	Yes
SignOn	Communication Sign On	SignOnGroup	Yes
Work Item List	Work Item List	WorkItemList	Yes
Working Time	Not applicable	Not applicable	Yes

Communications Toolbar Commands and Bitmaps

Table 22 on page 138 displays the names of the bitmap object definitions that are associated with the Siebel Tools commands for the communications toolbar items. From left to right, the table displays the same Siebel Tools command names shown in Table 21 on page 136, the name of the bitmap object definition specified for HTML Bitmap, and the name of the bitmap object specified for HTML Disabled Bitmap. HTML Bitmap is for controls in an enabled state, while HTML Disabled Bitmap is for controls in a disabled state.

The applicable image file names can be found in Siebel Tools, by viewing the bitmap object definitions for the bitmap category HTML Command Icons. The source image files are installed in the following locations:

- `webmaster\images\`*`language_code`*, in the Siebel Server installation directory
- `public\`*`language_code`*`\images`, in the Siebel Developer Web Client installation directory

where *language_code* represents the language code for the installed software, such as ENU for U.S. English.

NOTE: New or updated files on the Siebel Server are automatically populated to the Web server whenever the Web server (where the Siebel Web Server Extension is installed) is restarted. Files can also be manually updated from the Siebel Server without restarting the Web server. For more information, see *Siebel Security Guide* and the *Siebel Installation Guide* for the operating system you are using.

As noted, Table 22 displays the names of the bitmap object definitions that are associated with the Siebel Tools commands for the communications toolbar items.

Table 22. Communications Toolbar Commands and Bitmaps

Tools Command (Display Name)	HTML Bitmap	HTML Disabled Bitmap
Accept Incoming Email	Comm Email	Comm Email Disabled
Accept Miscellaneous Work Item	Comm Misc Work	Comm Misc Work Disabled
Accept Work Item	Comm Accept Work	Comm Accept Work Disabled
Answer Phone Call	Comm Phone	Comm Phone Disabled
Blind Transfer Work Item	Comm Blind Transfer	Comm Blind Transfer Disabled
Communication Sign On	Comm SignOn	Comm SignOn Disabled
Communication Sign Off	Comm SignOff	Comm SignOff Disabled
Consultative Conference Work Item	Comm Consultive Conference	Comm Consultive Conference Dis
Consultative Transfer Work Item	Comm Consultive Transfer	Comm Consultive Transfer Disab
Forward Work Item	Comm Forward Work	Comm Forward Work Disabled
Hold Work Item	Comm Hold Work	Comm Hold Work Disabled
Initiate Email	Comm Email	Comm Email Disabled
Initiate Fax	Comm Fax	Comm Fax Disabled
Initiate Phone Call	Comm Phone	Comm Phone Disabled
Initiate SMS	Comm SMS	Comm SMS Disabled
Initiate Work Item	Comm Initiate Work	Comm Initiate Work Disabled

Table 22. Communications Toolbar Commands and Bitmaps

Tools Command (Display Name)	HTML Bitmap	HTML Disabled Bitmap
InQueue Time	Comm InQueue Time	Comm InQueue Time
Media Indicator	Comm Phone	Comm Phone
Not Ready for Email	Comm Email	Comm Email Disabled
Not Ready for Phone	Comm Phone	Comm Phone Disabled
Not Ready State	Comm Not Ready	Comm Not Ready Disabled
Release Work Item	Comm Release Work	Comm Release Work Disabled
Resume Work Item	Comm Resume Work	Comm Resume Work Disabled
Retrieve Work Item	Comm Retrieve Work	Comm Retrieve Work Disabled
Work Item List	Not applicable	Not applicable

Modifying the Communications Toolbar

This topic describes how to modify the communications toolbar. It contains the following information:

- "Modifying the Function of an Existing Toolbar Button" on page 140
- "Modifying the Appearance of an Existing Toolbar Button" on page 141
- "Moving, Adding, or Removing a Toolbar Button" on page 142

In implementing Siebel Communications Server, you can modify communications toolbar functionality in several ways. Several types of modifications that might apply to your implementation are described in the linked topics.

For more information about configuring the communications toolbar in Siebel Tools, see the developer documentation on the *Siebel Bookshelf*.

NOTE: Generally, it is best not to make changes in Siebel Tools unless it is strictly necessary or it is done with other application development work.

Related Books

Configuring Siebel Business Applications

Configuring Siebel Open UI

Using Siebel Tools

Siebel Developer's Reference

Modifying the Function of an Existing Toolbar Button

You can modify the function of an existing communications toolbar button in several different ways. You can:

- Change which communications command an existing communications toolbar button invokes
- Change which device command an existing communications toolbar button invokes
- Change which business service method an existing communications toolbar button invokes

Modifying Communications Command for Existing Toolbar Button

If you created a command in the communications configuration and you want a communications toolbar button to invoke this communications command instead of an existing command, then rename your commands so that the new command uses the name of the command that is currently associated with the toolbar button.

Renaming commands in the communications configuration has less impact on an upgrade effort than modifying object definitions in Siebel Tools.

Renaming commands in the communications configuration can be used whether you are only implementing a custom command or you are both implementing a custom command and invoking a device command supported by a custom communications driver, as described in "Modifying Device Command for Existing Toolbar Button" on page 140.

For more information about communications commands, see "Commands" on page 116.

Modifying Device Command for Existing Toolbar Button

If you are using a custom communications driver, created using the Adaptive Communications API, in order to extend or replace the functionality of a driver provided with Siebel Business Applications, then you can reuse the existing communications toolbar configuration, and you can reuse existing commands from the configuration data provided with Siebel Business Applications. Of course, you can also create new communications commands, as described earlier in this topic.

If your communications driver supports different device command names than do the drivers provided with Siebel Business Applications (such as the Virtual CTI driver), then specify your own driver's device commands in your communications commands. Alternatively, for commands that do not specify a device command and do not execute something else such as a business service method or script, the application attempts to execute a device command with the same name as the communications command. For more information about device commands, see "Commands" on page 116. For more information about Adaptive Communications, see Appendix A, "Developing a Communications Driver."

Modifying Business Service Method for Existing Toolbar Button

If you have created a custom business service, and you want an existing communications toolbar button to invoke a method for this service, then you must use Siebel Tools to modify the command object definition that is associated with the toolbar item, to specify the new business service and method. This approach is of limited practical value, however, because you can customize the function of a button without using Siebel Tools.

For more information about how business service methods are specified for the toolbar buttons, see "Communications Toolbar Items, Commands, and Methods" on page 136.

Modifying the Appearance of an Existing Toolbar Button

You can modify the appearance of an existing communications toolbar button in several different ways. You can:

- Change the image file or bitmap object definition for an existing toolbar button
- Change the icon file for an interactive communications driver

Changing Image File or Bitmap Object Definition for Existing Toolbar Button

If you want to customize the communications toolbar by using your own graphical elements in place of existing elements, then you can do so by using your own graphics files in place of existing files.

You can change the image file specified for HTML Bitmap and the image file specified for HTML Disabled Bitmap. If you want to change the graphics used to represent a given channel, for example, then identify the files used to represent the channel and change them to use your new images. All toolbar buttons that use the same bitmap object definitions now use the new image files.

Suppose, however, that you want to change the appearance of a particular button, but you do not want to change the appearance of all buttons that display the same image file. In this case, you can create a new bitmap object definition in Siebel Tools, associate it with your new image file, and associate the toolbar item object definition with the new bitmap object definition.

For more information about how bitmaps are specified for the toolbar buttons and about the locations of the image files, see "Communications Toolbar Commands and Bitmaps" on page 137.

Changing the Icon File for Interactive Communications Driver

You can change the icon file for an interactive communications driver. The icon file represents communications work items of a particular channel. It is used for the Channel Type Indicator and to support blinking toolbar buttons.

Some communications toolbar buttons blink, based on event occurrence. For example, when a new work item arrives, the button for Accept Work Item blinks until the agent accepts the work item. When a work item has been paused, the button for Resume Work Item blinks until the agent resumes the work item.

For any buttons executing methods of the Communications Client business service, as do the default buttons on the communications toolbar, the blinking behavior is controlled by the applicable communications driver and cannot be configured. If a button is configured to execute a method of another business service, then this behavior is controlled by that service.

The names of the icon files are specified in the Icon File field in the Communications Drivers and Profiles view in the Administration - Communications screen. The files are in the same location as those for the bitmap object definitions.

Although different file names are used, the image content of the driver icon files is identical to those of some of the bitmaps used for the same communications channel. These bitmaps, and the locations of the associated image files, are described in "Communications Toolbar Commands and Bitmaps" on page 137.

For more information about interactive drivers and icon files, see "Interactive Drivers" on page 37.

Moving, Adding, or Removing a Toolbar Button

You can customize the communications toolbar to add a new toolbar button or to remove an existing button. You can:

- Move a toolbar button or other control on the communications toolbar

  You can move a toolbar control, in relation to other controls, by modifying the value of the Position field for the applicable toolbar item in Siebel Tools.

- Remove a toolbar button or other control from the communications toolbar

  You can prevent a control from appearing in the communications toolbar by setting the Inactive flag to True for the applicable toolbar item in Siebel Tools.

- Add a new button to the communications toolbar

  If you add a new toolbar button to the communications toolbar, then you must also add, or reuse, all associated object definitions in Siebel Tools and image files. The scope of what you have to do depends on the purpose of the new button.

Communications Toolbar Buttons and Commands

This topic provides an overview of the following:

- How communications commands are invoked from the communications toolbar
- About command parameters that affect toolbar commands
- About how group commands function to support the communications toolbar, especially in a multichannel environment

This topic contains the following information:

- "How Communications Toolbar Buttons Work" on page 143
- "Command Parameters Affecting Communications Toolbar Buttons" on page 143
- "Communications Group Commands in the Communications Toolbar" on page 144
- "ToolTip Text for the Communications Toolbar" on page 144

How Communications Toolbar Buttons Work

When an agent clicks a communications toolbar button, the Siebel client attempts to execute a command, corresponding to the method name, from the communications configuration.

If no matching command is found, then the communications driver (a profile for which is associated with the configuration) that supports a device command with the same name as the method executes that device command.

If a command invokes a subcommand, business service method, or script, then it does not have to execute a device command. All other commands generally execute a specified device command.

A communications command executing a device command can be explicitly associated with a profile for the driver that supports that device command. This can help to avoid possible collisions where the same device command is supported by multiple drivers and can also help the administrator to keep track of the intended functions of commands in a complex, multichannel environment.

For more information about the DeviceCommand parameter, see "Command Parameters" on page 119. For available device commands that might apply to your deployment, see:

- "Virtual CTI Driver Commands" on page 244
- "Special Commands for Device Commands" on page 87

Command Parameters Affecting Communications Toolbar Buttons

The presence and visual appearance of items in the communications toolbar are determined in part by a set of related command parameters defined for the communications commands that are accessible through the toolbar:

- **Description.** Provides the text to be displayed as ToolTip text for the toolbar button for the command. Many references to toolbar buttons by name are based on the ToolTip text specified with this parameter. For more information, see "ToolTip Text for the Communications Toolbar" on page 144.
- **Hidden.** Although this parameter has no direct effect on the communications toolbar, it is typically set to True for a command that is to appear in the communications toolbar (and that is not to appear in the Communications submenu).
- **LocalMenu.** Although this parameter has no direct effect on the communications toolbar, it is typically set to False, or omitted from the command, for a command that is to appear in the communications toolbar (and that is not to appear in the applet-level menu).

For more information about these parameters, see "Command Parameters Affecting Communications Menu Items" on page 145 and "Command Parameters" on page 119.

Communications Group Commands in the Communications Toolbar

In the communications configurations provided with Siebel Business Applications, commands whose names end in *Group* are commands that can contain several subcommands. Most of the communications toolbar buttons invoke group commands. A group command can invoke another group command.

When a group command is executed, such as might occur when the user clicks the associated toolbar button, one of its subcommands is typically invoked, based on the current context. Alternatively, you can configure a group command to execute all subcommands, where this is appropriate. The order you specify for each subcommand determines the sequence in which the subcommands are checked to find one that matches the context. For more information about group commands and subcommands, see "Hierarchical Commands (Commands and Subcommands)" on page 117.

ToolTip Text for the Communications Toolbar

The ToolTips displayed when the agent points to a communications toolbar button are usually derived from the command in the communications configuration. They might also be obtained from the communications driver, or from the associated command object definition in Siebel Tools.

The ToolTip text is derived in this fashion:

- For the communications command (which might be a subcommand of another command) that the toolbar button represents, the value of the command parameter Description is used as the ToolTip text. Obtaining text from the context-appropriate subcommand allows the ToolTip text itself to vary by context. For the MakeCallToContact command, for example, *Make Call to Contact* is the ToolTip text.
- If the command parameter Description is not used for the communications command that the toolbar button represents, then any ToolTip text defined in the communications driver for the associated device command is used. Such text might also be context-sensitive, depending on the driver implementation.

  NOTE: In order to update the description of a device command, a driver must use the client handle method CacheCommandInformation. In turn, this description is used for the ToolTip text.
- If neither of the previous two sources provides ToolTip text, then any ToolTip text is used that is defined for the command object definition in Siebel Tools. Such text cannot be context-sensitive, because only a single string is defined for the toolbar button.

Communications drivers and configurations provided with Siebel Business Applications provide ToolTip text. If you use such a driver, then ToolTip text derived from Siebel Tools is not used.

Configuring Communications Menu Commands

This topic contains the following information:

- Describes the Communications submenu and communications commands in the applet-level menu
- Explains how items in these menus relate to commands defined for the communications configuration
- Provides instructions and guidelines for modifying communications menu commands:
  - "Communications Submenu and Applet-Level Menu" on page 145
  - "Command Parameters Affecting Communications Menu Items" on page 145
  - "Communications Group Commands in Menus" on page 146
  - "Communications Menu Items and Device Commands" on page 147

For information about using the communications menu commands, see "Using Communications Menu Commands" on page 232.

Communications Submenu and Applet-Level Menu

When communications are enabled for a user, the Communications submenu is displayed in the Tools application-level menu. The Communications submenu is configured using commands defined in the communications configuration.

The name and position of *Communications* as an item in the Tools application-level menu is determined in Siebel Tools by the Tools - Communications menu item definition.

Like the communications toolbar, the exact content of the Communications submenu varies according to the communications configuration you are using. All the sample communications configurations provided with Siebel Business Applications contain definitions to configure the Communications submenu.

The applet-level menu, which is accessed at the top of each applet, below the applet name, is generally available in the Siebel client. When session communications are enabled, communications-related items are added to it, according to the commands defined for the communications configuration.

Command Parameters Affecting Communications Menu Items

The presence and visual appearance of items in the Communications submenu and applet-level menu are determined by a set of related command parameters defined for the communications commands that are to appear in the affected menus. For more information, see "Command Parameters" on page 119.

The following list describes the command parameters:

- **Hidden.** Set to False, or omitted from the command, for a command to appear in the Communications submenu. Set to True for a command that is intended to appear in the applet-level menu or the communications toolbar, or that is not to appear directly in the user interface.
- **HotKey.** Specifies the key combination that serves as a keyboard accelerator for the command's menu item.

  Keyboard shortcuts defined for communications commands do not use as input any data entered into the text entry field in the communications toolbar.

  NOTE: If you specify keyboard shortcuts, using the HotKey parameter, that conflict with keyboard shortcuts specified in Siebel Tools, then these shortcuts take precedence over the shortcuts from Siebel Tools for the agent's session.
- **HotKeyText.** Specifies locale-dependent text to represent the keyboard accelerator for the command's menu item. The keyboard accelerator itself is specified using the HotKey parameter. This locale-dependent text appears in the Communications submenu.
- **LocalMenu.** Set to True for a command to appear in the applet-level menu. Set to False, or omitted from the command, for all other commands.
- **MenuPosition.** Specifies the position of the command's submenu item, relative to the other submenu items in the Communications submenu, or the position of the command's menu item relative to other menu items in the applet-level menu. This parameter supports multiple menu levels.
- **Title.** Provides the text of the command's menu item in the Communications submenu or in the applet-level menu.

Communications Group Commands in Menus

In the communications configurations provided with Siebel Business Applications, the commands whose names end in *GroupInMenu* or *GroupInLocalMenu* might specify several subcommands, one of which is invoked when the menu item is chosen, based on the current context.

The order specified for each subcommand determines the sequence in which the subcommands are checked to find one that matches the context.

The *GroupInMenu* commands are for the Communications submenu, and the *GroupInLocalMenu* commands are for the applet-level menu.

For example, MakeCallGroupInLocalMenu has several subcommands defined, each of which specifies another command, such as MakeCallToAccount, MakeCallToContact, and so on. When the agent chooses Make Call in the applet-level menu, one of these subcommands is invoked, according to the current context.

If the agent is viewing a contact record, for example, then Contact is the current business component. Because this business component is specified in the command data defined for MakeCallToContact, a match is found in this command.

For more information about group commands and subcommands, see "Hierarchical Commands (Commands and Subcommands)" on page 117.

Communications Menu Items and Device Commands

When an agent chooses a communications command from the Communications submenu or the applet-level menu, the corresponding command is executed from the communications configuration. The communications driver (a profile for which is associated with the configuration) that supports the device command specified for the communications command executes that device command.

If a command invokes a subcommand, business service method, or script, then it need not execute a device command. All other commands generally execute a specified device command.

A communications command executing a device command can be associated with a profile for the driver that supports that device command. This can help to avoid possible collisions where the same device command is supported by multiple drivers and can also help the administrator to keep track of the intended functions of commands in a complex, multichannel environment.

For more information about the DeviceCommand parameter, see "Command Parameters" on page 119. For information about the commands supported by the Virtual CTI driver, see "Virtual CTI Driver Commands" on page 244. See also "Special Commands for Device Commands" on page 87.

Configuring Communications List of Values Types

This topic describes how to configure communication List of Values types. It contains the following information:

- "List of Values Type for Channel Type" on page 148
- "List of Values Type for ACD Queues" on page 148
- "List of Values Type for Reason Code" on page 149
- "List of Values Types for Event Parameters" on page 149
- "List of Values Types for Command Parameters" on page 150

Records defined for various List of Values types determine what values can be selected in certain fields (drop-down lists) in the Administration - Communications screen.

List of Values records of the same type and, where applicable, with the same Parent LIC (Language-Independent Code) value are listed together in particular drop-down lists. A List of Values record must also be set to Active in order to appear in the user interface.

Typically, List of Values records provided with Siebel Business Applications serve your needs without alteration. However, depending on your implementation, you might have to include custom content in a particular List of Values. To include custom content, you add a List of Values record, or modify an existing List of Values record to change its display value.

In particular, if you are implementing a custom communications driver, then you must create records for several List of Values types.

CAUTION: For an existing List of Values record, it is not recommended to modify default field values other than for the Display Value field. It is also not recommended to delete List of Values records provided with Siebel Business Applications. If you are certain that a List of Values record is not needed in any context, then you can uncheck the Active flag in order to hide it in the user interface. However, note that some List of Values records are required for internal operations not directly related to the user interface.

Some of the List of Values types for which you can provide your own values are described in the topics that follow. Your deployment might require you to add or modify records for additional List of Values types that are not described in this chapter.

To add or modify List of Values records, you use the Administration - Data screen and the List of Values Explorer and List of Values views.

NOTE: For more information about adding or modifying List of Values records, see *Siebel Applications Administration Guide*. Follow all documented guidelines and restrictions.

List of Values Type for Channel Type

This topic identifies the List of Values type for specifying the channel type for communications drivers.

If the available communications channel types do not meet your needs for your communications drivers, and you want to add or modify channels, then you must add or modify records to include the values you require. The applicable List of Values type is COMM_MEDIA_TYPE. The Parent LIC value is set to COMM for values to be displayed only in the Channel Type field in the Communications Drivers applet (in the Communications Drivers and Profiles view in the Administration - Communications screen). For more information about defining drivers, see "Configuring Communications Drivers and Profiles" on page 39.

List of Values Type for ACD Queues

If you are supporting the voice channel (using Siebel CTI), and supporting ACD queues for your agents, then you might require particular values to appear in the ACD Queue field in the ACD Queues applet. The ACD Queues applet appears in the Agent General Profile view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named CTI_ACD_QUEUES. For more information about specifying agents and ACD queues, see "Specifying Agents" on page 55.

List of Values Type for Reason Code

If you are supporting reason codes for your call or contact center agents to specify when they are not ready to receive communications work items, then you might require particular values to appear in the applet Transfer Multiple LOV Popup Applet. This applet appears when agents specify the Not Ready state for any or all supported communications channels. By default, this applet is multiselect enabled.

To specify your custom values, add or modify records for the List of Values type named REASON_CODE. For more information about how agents specify the Not Ready state, see "Using the Communications Toolbar" on page 223.

List of Values Types for Event Parameters

This topic identifies List of Values types for parameters for event handlers, event responses, and event logs.

NOTE: You must add List of Values records for these types only if you are using a custom communications driver.

For more information about specifying events, see "Defining Communications Events" on page 62.

List of Values Type for Event Handler Parameters

If you use a custom communications driver, then you require particular values to appear in the Name field (drop-down list) in the Event Handler Parameters applet. This applet appears in the All Event Handlers view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named COMM_EVTHNDLR_PARAM.

List of Values Type for Event Response Parameters

If you use a custom communications driver, then you require particular values to appear in the Name field (drop-down list) in the Event Response Parameters applet. This applet appears in the All Event Responses view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named COMM_EVTRESP_PARAM.

List of Values Type for Event Log Parameters

If you use a custom communications driver, then you require particular values to appear in the Name field (drop-down list) in the Event Log Parameters applet. This applet appears in the All Event Logs view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named COMM_EVTLOG_PARAM.

List of Values Types for Command Parameters

This topic identifies List of Values types that parameters for commands and command data use.

NOTE: You must add List of Values records for these types only if you are using a custom communications driver.

For more information about specifying commands, see "Defining Communications Commands" on page 67.

List of Values Type for Command Parameters

If you use a custom communications driver, then you require particular values to appear in the Name field (drop-down list) in the Command Parameters applet. This applet appears in the All Commands view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named COMM_CMD_PARAM.

List of Values Type for Command Data Parameters

If you use a custom communications driver, then you require particular values to appear in the Name field (drop-down list) in the Command Data Parameters applet. This applet appears in the All Command Data view, in the Administration - Communications screen.

To specify your custom values, add or modify records for the List of Values type named COMM_CMDDATA_PARAM.

7 Configuring Advanced Communications Features

This chapter provides information about advanced configuration for communications features, primarily for session communications. It includes the following topics:

Using Macro Expansion for Character Fields

Some parameter values for events or commands, communications drivers, or configuration parameters can contain macro-expansion characters, which allow the actual values of those fields to be calculated based on contextual data such as the current business component or other run-time data.

This topic contains the following information:

Macro-Expansion Syntax Elements

Macro expansion uses the following syntax elements:

- Brackets ([ ]) can contain a field name from the record that is currently selected in the Siebel application, for example, a record displayed in a dialog box. Brackets are applicable to event and command parameters only.
- Braces ({ }) can contain a field name from the current business object, business component, work item, or session data, or can contain one of the macros described in "Macros for Parameter Values" on page 152. The Communications Server processes the content of braces after executing a pop-up applet. This applies to command data only.
- A colon (:) following a field name functions as a separator to introduce modifying keywords, attributes, subfields, or numeric ranges that determine what data to extract and substitute. For information about how this element is used in phone number processing, see "Macro Expansion with Phone Numbers" on page 158.
- A backslash (\) functions as an escape character, allowing you to place a literal bracket, brace, or colon in a value without its being subject to macro expansion.

Macros for Parameter Values

Several macros (sometimes referred to as special fields) can be used within values for applicable event or command parameters, configuration parameters, or, in limited cases, driver parameters.

NOTE: Unless otherwise stated, you cannot use the macros described in this topic within values for communications driver parameters.

When the parameter value is read into memory (such as when the event or command is invoked), particular values are substituted for the macro name. These macros are not specific to any particular communications system, such as a specific CTI middleware product.

Some macros require work item attributes to be provided. Notations for specifying attributes are shown in examples. For more information, see "Work Item Attributes" on page 160.

Event attributes are also available for use in parameter values that support macro expansion. For example, the event attribute SiebelChannelType represents the channel type of the current work item. For more information, see "Driver Event Attributes" on page 266.

The macros that start with the at sign (@) are static variables with values that are defined at run time (values for some of these macros might change during a user session). The macros that start with the dollar sign ($) are actually functions that return a value. The macros, which have names that start with the at sign (@) or the dollar sign ($), are listed as follows:

- **@ACDDNList.** The list of ACD DNs (extensions of type *A*) associated with the current agent.

  The values are separated by commas.
- **@AgentId.** Agent login for the current agent.
- **@AgentPin.** Password for the agent login for the current agent.

- **@ClientHostName.** Host name of the agent's computer. You can use this macro with hoteling implementations using the Virtual CTI driver. For more information, see Chapter 10, "Using the Virtual CTI Driver."
- **@ClientIP.** IP address of the agent's computer. You can use this macro with hoteling implementations using the Virtual CTI driver. For more information, see Chapter 10, "Using the Virtual CTI Driver."
- **@Configuration.** The current configuration for the agent.
- **@CountryCode.** The country code applicable to the agent's location.

  If the agent's country is the U.S., then this macro returns an empty value. If the agent's country is other than the U.S., then the macro returns the applicable country code, preceded by a plus (+). For example, if the agent's country is France, then the macro returns +33.
- **@DeselectedWorkItem.** The item in the Work Items list in the communications toolbar that had the focus just prior to the item that currently has the focus. (The item with the focus is represented by the @SelectedWorkItem macro.) This macro must be specified with an attribute value, as follows:

  @DeselectedWorkItem:*attribute_name*

  where *attribute_name* is the work item attribute you are providing. For more information, see "Work Item Attributes" on page 160.
- **$DialingRuleMethod.** Used within parameter values for the DialingFilter.Rule*N* configuration parameters, this macro calls a custom business service method to determine how to translate phone numbers when the dialing filter is in effect.

  The custom business service must recognize the input arguments Filter and PhoneNumber. Filter is the filter in effect for the parameter that invokes the macro. PhoneNumber is the number that needs to be translated. For example, for the following dialing rule, Filter is 650:

  DialingFilter.Rule2 = "650->$DialingRuleMethod(*myservice*.*mymethod*)"

  The business service must also support an output parameter called PhoneNumber. For example, if your dialing filter rule is to be applied to the U.S. number (650) 123-4567, where numbers with prefix 123 in the area code (650) must be changed to use prefix 555, then the business service method might return the PhoneNumber output parameter with the value *916505554567*.

  After a dialing filter rule is applied that uses the macro $DialingRuleMethod, no other filters are applied.

  For more information about using this macro with dialing filters, see "Working with Dialing Filters" on page 165.
- **@DNList.** The list of DNs (standard extensions of type *S*) associated with the current agent.

  The values are separated by commas.
- **@EditControl.** The data in the edit field (text box) control in the communications toolbar.

  This macro yields a value only when the edit field in the communications toolbar has the focus and contains a value. Use @EditControl when agents input values other than phone numbers into the communications toolbar's edit field, such as to log in to the switch.

- **$GetCommandStatus.** Obtains the status of the device command for a communications command. Possible status values returned are:
  - **Checked.** The command is in toggled-down state, and the button displays (for example, if the agent indicates Not Ready for new work items, then the button is toggled down).
  - **Blinking.** The command is available, and the button is blinking.
  - **Enabled.** The command is available, and the button is enabled.

    NOTE: You can use this macro within command parameters, but not within event parameters. Events cannot use this macro to determine the status of a device command.
  - **Disabled.** The command is not available, and the button is disabled.

  You can obtain the command status with an expression like this:

  `$GetCommandStatus(`*`device_command`*`)`

  where *device_command* is the name of the device command for the command. For example, a command parameter to determine the status of the device command might be defined as follows:

  FilterSpec = "[$GetCommandStatus(ChangeNotReadyState)]='Checked'"

  In this case, if the status of ChangeNotReadyState is Checked, then FilterSpec matches, and the command containing this FilterSpec parameter can execute.

- **$GetInboundWorkItemAttr.** Obtains the attributes of an inbound work item.

  You can obtain the work item attributes with an expression like this:

  $GetInboundWorkItemAttr(*channel_name*, *attribute_name*)

  where *channel_name* is the name of the locale-independent channel for the work item, and *attribute_name* is the work item attribute you are interested in. For example, a command data parameter to obtain the tracking ID of an inbound voice work item might be defined as follows:

  Param.TrackID = "{$GetInboundWorkItemAttr(Voice, DriverWorkTrackID)}"

  For more information, see "Work Item Attributes" on page 160.

- **$GetWorkItemAttr.** Obtains the attributes of a work item, for example, in an event handler evaluating a new work item, or in a command.

  You can obtain the work item attributes with an expression like this:

  $GetWorkItemAttr(*workitem_ID*, *attribute_name*)

  where *workitem_ID* is the ID number of the work item, if known, and *attribute_name* is the work item attribute you are interested in.

  For example, an event handler parameter to verify that the ContactId attribute (a custom work item attribute) has a value might be defined as follows:

  FilterSpec = "[$GetWorkItemAttr(SiebelWorkItemID, ContactId)] IS NOT NULL"

  For commands operating on the selected work item, you can also obtain work item attributes using @SelectedWorkItem:*attribute_name*.

For more information, see "Work Item Attributes" on page 160. For more information about the SiebelWorkItemID attribute, see "Driver Event Attributes" on page 266.

- **$HotelingPhone.** Retrieves an agent's extension if the agent is using a hoteling teleset. The field value is calculated at run time according to these rules:
  - Uses the login name of the currently selected agent (employee), or the name of a service request owner, to find the agent's run-time extension, if the agent is using a hoteling teleset. In the first example following, *Login Name* is the Login Name field of the current business component, which corresponds to the LOGIN column in the S_USER table.
  - If the extension is not found, then $HotelingPhone retrieves the employee's extension number from the S_USER table.

  The following example parameter definition in a make-call command (command data definition) might generate the extension you want, based on the Login Name field:

  ```
  Param.PhoneNumber = "{$HotelingPhone(Login Name):Lookup}"
  ```

  Alternatively, the following example parameter definition for a make-call command might generate the extension you want, based on the Owner field. Use a parameter value like this for a command invoked when the current business component is Service Request, to call the owner of the current service request record.

  ```
  Param.PhoneNumber = "{$HotelingPhone(Owner):Lookup}"
  ```

  For more information, see "Configuring Telesets for Hoteling" on page 168.

  NOTE: For a deployment using the Virtual CTI driver, this topic does not apply. However, you can implement hoteling using this driver. For more information, see Chapter 10, "Using the Virtual CTI Driver."

  NOTE: The macro @HotelingPhone is still supported, for compatibility with release 6.*x*, but it is recommended that you use $HotelingPhone instead.

- **@Language.** The language code applicable to the agent's Siebel application session. For example, the language code for U.S. English is *ENU*.
- **@Now.** Returns the current timestamp, using the following format:

  ```
  %month/%day/%Year %H:%M:%S
  ```

- **@Phone.** Represents a phone number, whose value is calculated at run time according to these rules:
  - If the Phone # field in the communications toolbar has the focus and contains a value, then @Phone is equal to the value of this field.
  - If the Phone # field has no value, but the currently active applet field is of the type DTYPE_PHONE, then @Phone is equal to the value of this field.
  - If the Phone # field has no value and the currently active field is not of the type DTYPE_PHONE, then @Phone is equal to the value of the field referred to by the Primary Phone Field user property for the business component.

  In all other cases, the @Phone macro contains no data. For more information, see "Macro Expansion with Phone Numbers" on page 158.

- **@PrimaryQueueList.** The list of ACD queues associated with the agent and designated as primary. For example, a login command might include a parameter like this to identify which ACD queues to log the agent into:

  ```
  Param.ACDQueue = "{@PrimaryQueueList}"
  ```

  The values are separated by commas.

- **@QueueId.** The list of ACD queues associated with the agent and designated as primary.

  NOTE: @PrimaryQueueList replaces @QueueId, but you can still use @QueueId for compatibility purposes.

- **@QueueList.** The list of ACD queues associated with the agent. This list includes all such queues, including those marked as primary and those not marked as primary. The values are separated by commas.

- **@Random.** A string containing 10 random digits. You can use this macro for testing purposes.

- **$RemoteConnectStr.** The name of the remote call center. This macro, which you can use with transfers and conference calls between call centers, derives the name of a remote call center's communications configuration from either:

  - The ConnectString configuration parameter (if defined), or
  - The extension number of the agent to be called

  For example, a command parameter in a transfer or conference command might generate the configuration name in this fashion:

  ```
  Param.RemoteConnectStr = "[$RemoteConnectStr(@Phone)]"
  ```

  For more information, see "Configuring Remote Transfers and Conferences" on page 175.

- **$RemoteConnectStr2.** The name of the remote call center. This macro, which you can use with transfers and conference calls between call centers, derives the name of a remote call center's communications configuration from either:

  - The ConnectString configuration parameter (if defined), or
  - The employee ID of the agent to be called

  For example, where the SelectBusObj and SelectBusComp parameters are set to *Employee*, a command parameter in a transfer or conference command might generate the configuration name in this fashion:

  ```
  Param.RemoteConnectStr = "[$RemoteConnectStr2(Id)]"
  ```

  The Employee business component uses the Id field to uniquely identify an employee record. A command that references another business component might have to use another field name to uniquely identify a record. For more information, see "Configuring Remote Transfers and Conferences" on page 175.

- **@SelectedDN.** The currently selected extension in the Communications options of the User Preferences screen.

- **@SelectedQueue.** The currently selected ACD queue record in the Agent Queues list in the Communications options of the User Preferences screen.

- **@SelectedText.** Text the user has selected in the browser, such as in a field in an applet within the Siebel application. The text might be used, for example, to provide a phone number to call or to provide some other input data for a command.
- **@SelectedWorkItem.** The item in the Work Items list in the communications toolbar that has the focus. (The item that had the focus just prior to this item is represented by the @DeselectedWorkItem macro.) This macro must be specified with an attribute value, as follows:

  ```
  @SelectedWorkItem:attribute_name
  ```

  where *attribute_name* is the work item attribute you are providing. For more information, see "Work Item Attributes" on page 160.
- **@UserId.** Siebel login ID of the current Siebel user.
- **@UserName.** Login name of the current Siebel user.
- **@UIPhone.** The phone number that the agent clicked. See also the description of the communications parameter CallFromUICommand, in "Parameters for Communications Configurations" on page 47.
- **@WorkDuration.** Length of time, in seconds, of the current work item. That is, the time elapsed since the work item started.
- **@WorkObjectID.** Row ID of the work-item tracking object that was transferred to the agent or that was created as a result of creating an event log and defining a value for the AfterWork event log parameter.
- **@WorkStartTime.** Date and time (timestamp) when the work item arrived or was initiated. The timestamp is in the following format:

  ```
  %month/%day/%year %H:%M:%S
  ```

@Phone Macro Example

The example command and associated command data shown in Table 23 and Table 24 use the @Phone macro, which you can use to dial any phone field on any business component.

Table 23. Command: MakeCallToCurrentPhone

Parameter Name	Parameter Value
Description	Make Call to "{@Phone}"
DeviceCommand	MakeCall
Hidden	True

Table 24 shows a command data definition for this command definition.

Table 24. Command Data: MakeCallToCurrentPhone

Parameter Name	Parameter Value
AttachContext	True
Param.CallNotifyText	Call from {@UserName}...
Param.PhoneNumber	{@Phone:PhoneTypeLookup}
RequiredField.@Phone	?*

Macro Expansion with Phone Numbers

Macro expansion provides many options for retrieving and manipulating phone number data for use with communications events or commands. This topic applies to event and command parameters only.

The macro @Phone is often used to retrieve a phone number; for a detailed description of this macro and its encoded logic, see "Macros for Parameter Values" on page 152. Alternatively, you can explicitly specify the name of a field from which to extract phone number data.

NOTE: When you work with phone number fields in event and command parameter definitions, you must be aware of how phone number data is formatted in your deployment of the Siebel Business Applications and of the implications of the formatting. Dialing filters must be specifically designed and implemented to accommodate local requirements for transforming phone number data from the stored format. For phone number data stored in fields using the DTYPE_PHONE data type, when matching phone number data through QuerySpec or QuerySpec2 event response functions, you might have to use LIKE rather than = (equals) in predicate clauses.

For information about phone number formatting, including formatting for international numbers, see *Siebel Applications Administration Guide*.

You can use the following special macro-expansion features when retrieving phone number data:

- A keyword to specify how to apply dialing filters to the field value. For more information about the keywords, see "Keywords to Specify Dialing-Filter Behavior" on page 158.
- A keyword to indicate the part of the phone number that you want to extract. For more information about the keywords, see "Keywords to Extract Part of a Phone Number" on page 159.
- A specific range of numerals to extract from the field value. For more information about the ranges, see "Numeric Ranges to Extract" on page 160.

Keywords to Specify Dialing-Filter Behavior

Follow the field name with a colon (:) and one of the following keywords to specify whether to filter the field data using dialing filter rules specified using DialingFilter.Rule*N* configuration parameters. Dialing filters are applied in numeric sequence until a match is found.

Dialing filter rules allow you to perform an intelligent translation of a phone number, in order to optimize dialing. You can use the macro $DialingRuleMethod within a dialing filter rule definition to invoke a business service method to specify custom logic in the application of the dialing filter rule to the results.

You can use the following keywords with either the @Phone macro or a field whose name you explicitly specify:

- **Lookup.** Applies the dialing filters to the field data.

  You can use Lookup with fields that you know to contain phone number data that can be successfully filtered with your dialing filters.

- **PhoneTypeLookup.** Checks if a field is of type DTYPE_PHONE (as defined in Siebel Tools). If it is of this type, then the dialing filters are applied to the field data. If it is not of this type, then the dialing filters are not applied to the field data.

CAUTION: In order for phone number data to be handled correctly in your application, do not modify the type for any field defined as DTYPE_PHONE.

For examples, see "Macro-Expansion Examples" on page 164. For more information about DialingFilter.Rule*N* configuration parameters, see "Specifying Parameters for a Communications Configuration" on page 46.

Keywords to Extract Part of a Phone Number

Follow a field name with a colon (:) and one of the following keywords to specify the portion of the phone number field to be extracted and substituted. You can use these keywords only with fields of type DTYPE_PHONE. If none of these keywords is specified after a field name, then the phone number from the field corresponds to IntlNumber. IntlNumber can also be specified explicitly.

NOTE: The plus sign (+) is included only in the value for IntlNumber, when this keyword represents an international number. In the following keyword descriptions, references to the country code do not include the plus sign (+) character.

- **IntlNumber.** Represents *one* of the following:
  - If the country code for the number matches the current locale for the Siebel Server (Application Object Manager), then IntlNumber represents the domestic phone number (number without the country code or extension).
  - If the country code for the number does *not* match the current locale for the Siebel Server (Application Object Manager), then IntlNumber represents the international phone number (+ and country code and number, without the extension).
- **Digits.** The full phone number as stored in the database field (country code, phone number, and extension).
- **Country.** The country code only (without the phone number or extension).
- **Number.** The phone number only (without the country code or extension).
- **Extension.** The phone number extension only (without the country code and phone number). This keyword is applicable when the phone number field contains an extension number prefaced with an *x* or another localized character).

Numeric Ranges to Extract

Follow the field name or a keyword with a colon (:) and a range of numerals to be extracted from the field value. You can specify a substring numeric range directly after the field name or after a keyword indicating the part of the phone number.

Primary Phone Field User Property in Business Components

Any business component can have a named user property called Primary Phone Field. The property value is the field name of the same business component that contains the phone number and that must be dialed when a make-call command is invoked on an applet that uses this business component.

For example, the Account business component can have a Primary Phone Field user property set to the value Main Phone Number. This tells Communications Server that Main Phone Number is the field that must be dialed by default when dialing is requested on an account.

Under certain circumstances, the value of the @Phone macro is equal to the field designated as the Primary Phone Field.

For more information about specifying user properties, see the developer documentation on the *Siebel Bookshelf*.

Related Books

Configuring Siebel Business Applications

Configuring Siebel Open UI

Using Siebel Tools

Siebel Developer's Reference

Work Item Attributes

Table 25 on page 161 describes work item attributes that you can specify in values for parameters that support macro expansion. Communications events and commands can reference these attributes as subparameters of the parameter Param, or as elements within other parameter values where macro expansion is supported. The following macros, when used in parameter values, can process these attributes: @SelectedWorkItem, @DeselectedWorkItem, and $GetWorkItemAttr.

You can use the WorkTrackingObj parameter in event log or command data definitions to store or update custom work item attributes. You can use the AfterWork event log parameter to update database records that store work item attributes after work items are released. (You can also use the WorkObject event response parameter to update database records for tracking work items.)

For example, if an event log includes a parameter definition like the following, then *ContactId* becomes a custom work item attribute that is available for use wherever work item attributes can be accessed. For example, commands that update the customer dashboard can access such attribute values:

```
WorkTrackingObj.ContactId = "{Contact.Id}"
```

Alternatively, if an event log includes a parameter definition like the following, then text you specify is displayed in the Work Items list in the communications toolbar:

```
WorkTrackingObj.Description="your descriptive text goes here"
```

NOTE: Of the predefined work item attributes described in Table 25 on page 161, only ParentWorkItemID and Description can be updated.

For more information about the WorkTrackingObj and AfterWork event log parameters, see "Event Logs" on page 113. For more information about the WorkTrackingObj command data parameter, see "Command Data" on page 128. For more information about the WorkObject event response parameter, see "Event Responses" on page 100. For additional attributes that you can specify in event handlers, see "Driver Event Attributes" on page 266.

Table 25. Work Item Attributes

Attribute Name	Description
ChannelType	The language-independent value representing the channel type of this work item.
ChannelTypeLocale	The locale-dependent value for the channel type of this work item.
Description	A description of this work item. The value for this attribute comes primarily from the client handle methods WorkItemStarted and IndicateNewWorkItem. For more information, see Appendix A, "Developing a Communications Driver."
DriverWorkTrackID	The ID number of this work item as tracked by the driver.
IsRevoked	Indicates whether the work item has been rejected by an agent (using the Accept or Reject dialog box). Possible values are True or False (the default is False). This attribute is True whenever the service handle method RevokeQueuedWorkItem has been invoked for the work item.

Table 25. Work Item Attributes

Attribute Name	Description
ParentWorkItemID	The ID number of the parent of this work item. Note the following behavior for parent and child work items: ■ When a parent work item is released, all child work items are released. ■ When a parent work item is suspended, all child work items are suspended. ■ When a parent work item is resumed, all child work items are resumed. ■ When a child work item is started or resumed, the parent work item gains the focus. ■ Agent can use the Work Items list in the communications toolbar to select a child work item and operate on this child work item individually. ■ Parent and child work items can be of different channels, such as when an outbound voice call is placed to a customer who initiated an email work item. For more examples, see the sample communications configurations provided with Siebel Business Applications.
ProfileName	The communications driver profile for which this work item is applicable.
ViewBookmark	The bookmark for the view which was current when this work item was suspended. The bookmark data is serialized and compressed into a string.
WorkDuration	The duration of this work item. The value is the same as the @WorkDuration macro.
WorkItemID	The ID number of this work item.
WorkObjectID	The work tracking object ID number. The value is the same as the @WorkObjectID macro.

Table 25. Work Item Attributes

Attribute Name	Description
WorkStartTime	The start time of this work item. The value is the same as the @WorkStartTime macro. The value for this attribute comes primarily from the client handle method WorkItemStarted. For more information, see Appendix A, “Developing a Communications Driver.”
WorkState	The state of this work item. Possible values are Created, Active, Suspended, and Released. The value for this attribute comes primarily from client handle methods. ■ When IndicateNewWorkItem is invoked, the WorkState value is Created. ■ When WorkItemStarted is invoked, the WorkState value is Active. ■ When WorkItemSuspended is invoked, the WorkState value is Suspended. ■ When WorkItemResumed is invoked, the WorkState value is Active. ■ When WorkItemReleased is invoked, the WorkState value is Released. For more information, see Appendix A, “Developing a Communications Driver.”

Macro-Expansion Examples

The examples in Table 26 on page 164 demonstrate macro-expansion usage. From the nominal parameter value shown in the second column, macro expansion calculates the resulting value shown in the third column. Each example is explained to show how the result was generated.

Table 26. Macro-Expansion Examples

Parameter Name	Parameter Value	Result	Explanation
Param.PhoneNumber	#8[Phone #: 7-10]	#85000	If the value of the Phone # field in the current business component is 6505065000, then macro expansion extracts the seventh through tenth digits of the phone number. The characters *#8* prefix the result. Because no keyword is specified, the Number part is assumed, and the country code and extension are excluded. Because the macro-expansion characters representing the phone number are enclosed in brackets, this Phone # field can be in a dialog box or in the current business component.
Param.PhoneNumber	{Phone #: Number:6-10}	75000	Assume the same value for the Phone # field as in the previous example. Macro expansion extracts the sixth through tenth digits of the phone number. In this example, the Number keyword is explicitly specified. Because the macro-expansion characters representing the phone number are enclosed in braces, this Phone # field cannot be in a dialog box.
Param.GetId	{Id}	10-CSAE	If the Id field for a record in the current business component is 10-CSAE, then macro expansion yields this string as the result.
Param.Greet	\{Hi\}	{Hi}	The backslashes allow the braces to be included literally as part of the result. As a result, *Hi* is not interpreted as a field name.

Table 26. Macro-Expansion Examples

Parameter Name	Parameter Value	Result	Explanation
Param.ExtField	{Extension}	5000	For a field called Extension, macro expansion extracts the value of this field, for example, 5000. Because the Extension field is not of type DTYPE_PHONE, you cannot use keywords to specify a phone number part to extract.
Param.Ext	{Work Phone #: Extension}	6000	If the value of the Work Phone # field of the current business component (for example, Contacts) is (650)5065000 x6000 for a U.S. phone number and extension, then only the extension number after the *x* is extracted.
Param.PhoneNumber	{Work Phone #: Lookup}	*phone num*	Macro expansion generates a phone number based on applying the dialing filters to the data in the Work Phone # field of the current business component.
Param.PhoneNumber	{@Phone: PhoneTypeLookup}	*phone num*	Macro expansion generates a phone number based on the logic described for the @Phone macro, in "Macros for Parameter Values" on page 152. Dialing filters are applied only if a field is of type DTYPE_PHONE.

Working with Dialing Filters

Dialing filters are used by the Siebel application to manipulate telephone numbers for voice calls made, transferred, or for conference calls made.

Dialing filters specify a set of phone-number translation rules that are invoked when the Lookup or PhoneTypeLookup keyword is specified in macro-expanded text in a communications command for the voice channel (using Siebel CTI). Dialing filters are defined as configuration parameters in the All Configurations view in the Administration - Communications screen.

In each of the examples in this topic, the first set of numbers is searched for. If there is a match, then the searched numbers are translated to the numbers after the -> symbols (hyphen and right angle bracket).

Filter rules are checked for matches in numerical sequence. The last rule in the sequence must always be defined to match any number.

The numeric elements in the names of the filter rule parameters are applied by string comparison, from small to large. For example, this means that a dialing filter named DialingFilter.Rule11 is applied *before* one named DialingFilter.Rule2. Consequently, use the same number of digits for each rule, according to your total anticipated number of rules. For example, the numeric elements in the filter rules might use a sequence starting with 000, followed by 001, and so on, up to 999.

NOTE: How dialing filters actually function is subject to the command data definition for the command that is performing the make-call operation.

By default, the Lookup and PhoneTypeLookup keywords used in command data definitions do the following:

- Treat a phone number as domestic (that is, omit the country code) if the country code matches the current locale.
- Treat a phone number as international (that is, include the country code) if the country code does not match the current locale.

For Siebel CTI implementations supporting multiple call centers, consider the dialing filter requirements among the factors in deciding how to organize your agents in communications configurations. You might decide, for example, to ensure that all agents included in a single configuration are located within a single location for which the same set of dialing filters applies. For example, this grouping might be defined as a particular call center facility, or a particular locale corresponding to a country code.

For more information about configuration parameters, including DialingFilter.Rule*N*, see “Specifying Parameters for a Communications Configuration” on page 46.

For more information about the Lookup and PhoneTypeLookup keywords, keywords to represent parts of phone numbers, phone-number formatting, and the $DialingRuleMethod macro, see “Using Macro Expansion for Character Fields” on page 151.

For more information about international telephone numbers, see *Siebel Applications Administration Guide*.

Dialing Filter Examples

Table 27 on page 167 provides examples of dialing filter rules.

Table 27. Dialing Filter Examples

Dialing Filter	Description
DialingFilter.Rule001 = "650506->"	This rule takes a ten-digit domestic phone number (for example) and translates it into four digits for dialing an internal extension. You can use such a rule for internal calls, call transfers, or call conferences, where, for example, the U.S. area code 650 and prefix 506 apply to the call center. Or use such a rule for calls, transfers, or conferences to another call center, where this area code and prefix apply to the destination call center. For more information, see "Configuring Remote Transfers and Conferences" on page 175.
DialingFilter.Rule002 = "650->9"	This rule takes a ten-digit domestic phone number (for example) and translates it into eight digits for dialing an outside local call in the U.S.
DialingFilter.Rule003 = "+33->901133"	This rule takes a French phone number (any number for which the country code is 33, when the current locale is *not* France) and prefaces it with the digits that are required for dialing an international number when your locale is the U.S. or Canada.
DialingFilter.Rule004 = "+->9011"	This rule takes any international phone number (that is, any number for which the country code does not match the current locale) and prefaces it with the digits required for dialing an international number when your locale is the U.S. or Canada.
DialingFilter.Rule005 = "->91"	This rule takes any domestic phone number (that is, any number that did not match the preceding rule) and prefaces it with the digits required for dialing an outside call to another area code in the U.S.

For the examples in Table 27 on page 167, note the following:

- If a telephone number matches 650506****, then Rule001 is applied.
- If the number does not match this pattern, but matches 650*******, then Rule002 is applied.
- If the number does not match this pattern, but matches a French number (when the current locale is not France), then Rule003 is applied.
- If the number does not match this pattern, but matches any international number (for which the country code does not match the current locale), then Rule004 is applied.

- If the number does not match any of these patterns, then Rule005 is applied. As noted, the last defined rule must match any number.

The following example is an alternative definition for DialingFilter.Rule002.

```
DialingFilter.Rule002 = "650->$DialingRuleMethod(service.method)"
```

This rule takes a ten-digit domestic phone number (for example) and translates it into a number that is determined by the specified business service method. You can use such a rule to invoke a method that provides a correct phone number to dial, where other filter rules might not be able to filter unambiguously. For example, if internal numbers use a prefix, such as 506, that is also used outside the company, then the business service invoked using this macro can determine the correct number to dial by identifying the specific number as internal or external.

The following example is an alternative definition for DialingFilter.Rule004.

```
DialingFilter.Rule004 = "+->$DialingRuleMethod(service.method)"
```

This rule takes an international phone number and translates it into a number that is determined by the specified business service method.

Configuring Telesets for Hoteling

You can configure the telesets in your call center for *hoteling*, which allows an agent to log in to the Siebel application from any one of a pool of telesets and computers that have been configured for this purpose and allows the agent to use voice communications features.

NOTE: For a deployment using the Virtual CTI driver, this topic does not apply. However, you can implement hoteling using this driver. For more information, see Chapter 10, "Using the Virtual CTI Driver."

You configure hoteling by associating a teleset with the host name of the computer that is located at the same station as the teleset. Do this in the All Telesets view in the Administration - Communications screen. For more information about configuring telesets, see "Specifying Telesets" on page 58.

If you have specified a host name to enable hoteling for a teleset, then any agent logging in to the Siebel client on the hoteling computer uses the standard extension of the hoteling teleset that is associated with that computer's host name.

Hoteling Requirements and Issues

Note the following requirements and issues for implementing hoteling:

- After each agent is added to a communications configuration, the agent is typically associated with a teleset. However, associating a computer's host name with a teleset overrides this association and allows that computer to be used by multiple agents for hoteling.
- If an agent logs in to the Siebel application on a computer that is not associated with a hoteling teleset, then voice communications are enabled for the agent only if that agent is explicitly associated with the teleset at the new location.

- You can configure hoteling only for computers that are always expected to be located at the same station as the hoteling teleset. For example, although you can configure hoteling for a laptop computer that an agent connects to the network at any of several locations, you can configure it only for a single location where a particular teleset is located.
- Agents who log in to a hoteling computer cannot specify any other teleset in the Communications options of the User Preferences screen. The agent can specify a different extension for this teleset, however, if more than one extension is defined.
- For internal calls to employees who use hoteling telesets, you can configure commands to retrieve the run-time extension of the employee by using the $HotelingPhone macro. For more information, see "Macros for Parameter Values" on page 152.
- For agents using Siebel Open UI applications with proxy connections enabled in the browser to be able to use hoteling, you must set the parameter AutoLoadDriver to False. This setting allows agents to manually enter the computer's host name and to specify that they are using hoteling. For more information, see the description of the AutoLoadDriver parameter in "Parameters for Communications Configurations" on page 47.

Configuring Multitenancy

Siebel Communications Server supports the multiple-organization feature in the Siebel Business Applications. This feature, called *multitenancy* for call centers, helps provide the contact center, including those supporting multiple channels, with flexibility to organize its work and agent resources and to control visibility of data.

Contact centers that provide outsource services to other companies, or to multiple internal entities, can support these multiple entities using multitenancy.

In an outsource contact center, agents with expertise supporting products or services from Company A, for example, are assigned positions in the organization defined for Company A, while agents supporting Company B are assigned positions in the organization defined for Company B.

You can assign agents to multiple positions in multiple organizations, according to their expertise and the contact center's operational requirements.

For more information about positions, responsibilities, and organization visibility for Siebel applications, see *Siebel Security Guide* and *Siebel Applications Administration Guide*.

This topic contains the following information:

- "Setting the MultiTenancy Configuration Parameter" on page 169
- "Organization Visibility and Positions" on page 170

Setting the MultiTenancy Configuration Parameter

The Siebel Communications Server configuration parameter MultiTenancy allows you to specify whether organization-visibility rules must apply:

- Set this parameter to True if your Siebel implementation uses multitenancy. The parameter is set to True by default.

- Set this parameter to False if your Siebel implementation does not use multitenancy.

For more information, see "Specifying Parameters for a Communications Configuration" on page 46.

Organization Visibility and Positions

Some Siebel data records are visible only to users whose current position is within a particular organization. Other records are visible to users whose positions are in different organizations. You can associate records with one or more organizations.

Each user can have only one position active at a time. In some situations, a user might be allowed to see data for all of that user's assigned positions.

Organization visibility is defined for both business components and views.

Organization Visibility for Business Components

Organization visibility is defined for each business component, determining which of the following applies to the business component:

- Visibility is for one organization.
- Visibility is for multiple organizations.
- Organization visibility does not apply.

A business component for which organization visibility applies is said to be position-dependent.

Organization Visibility for Views

Organization visibility is enforced for each view. Multiple views that display data for the same business component might enforce visibility differently. For example, in the Contacts screen, the All Contacts view allows a user to view contact records within one organization, while the All Contacts across Organizations view (not available for all users) allows a user to view all contact records for all organizations.

You can determine whether a view enforces organization visibility by using Siebel Tools. For each view object definition:

- Organization visibility is enforced for the view if the field Visibility Applet Type is set to Organization *and* the field Admin Mode Flag is not checked (False).
- Organization visibility is *not* enforced for the view if the field Admin Mode Flag is checked (True).

Changing Position Manually or Automatically

For the agent to view some data, such as for certain screen pops, the agent's position must be changed, either manually or automatically. At any time, an agent can manually change the current position in the Siebel application by using the Change Position view in the User Preferences screen.

If the agent is the recipient of a screen pop for an incoming or transferred work item or a conference call, then the agent's position is changed automatically, as appropriate for the context and for the screen-pop data. When an agent retrieves a paused work item, the original screen pop and position are both restored. As described earlier, the business component determines the organization visibility for the screen-pop data, enabling the agent's position to be changed, as necessary.

What screen-pop data to display is determined by the current business component, which is specified in the QueryBusComp parameter in the event response. For more information about event responses, see "Event Responses" on page 100.

After an agent's position has been automatically changed, when a work item is concluded, the agent's position remains what it was changed to. The agent must manually change the position again if the new current position is not appropriate.

Scenarios for Generating Screen Pops and Changing Position

When a position is changed, a message is displayed to the user. If the position is not changed, then no message appears, except in certain error conditions. Application behavior for generating screen pops and changing positions varies for each of the scenarios is described as follows:

- If the agent's current position matches the screen-pop data, then the screen pop is displayed and the agent's position is not changed.
- If the agent's current position does not match the screen-pop data, and the agent has one other position that matches the data, then the screen pop is displayed and the agent's position is changed.
- If the agent's current position does not match the screen-pop data, and the agent has more than one position that matches the data, then the screen pop is displayed. The agent's position is changed to the first position that matches the data, and the agent is allowed to view all data that is defined to be visible for all the agent's assigned positions.

  The Siebel client allows the agent to view this data by using the VIEW_ALL mode, which ignores organization-visibility rules. A message advises the agent to manually change to a position appropriate for the screen-pop data, so that the agent can subsequently navigate to related records without hindrance.
- If the agent does not have a position that allows the agent visibility to the screen-pop data, then no screen pop is displayed.
- If the business component for a view does not have enough data to determine organization visibility, then the screen pop occurs without restriction. The position is not changed. No error message is displayed.
- Organization-visibility rules for screen pops apply to inbound work items and to transfers and conferences between contact-center users. For inbound work items, the Siebel application always attempts to set the user's position automatically. For transfers and conferences, the position is set only if the screen-pop view enforces organization visibility.
- When a view has no records displayed, organization-visibility rules do not apply to screen pops for call transfers or conferences.
- When organization visibility is not enforced for a view, organization-visibility rules are not used when changing an agent's position.

- Multitenancy affects screen pops differently for different views. Siebel application developers, contact-center managers, and end users must understand how this issue relates to screen pops.

  Some views, such as administration views, do not have organization-visibility rules. Some views, such as views whose names include the phrase *Across Organizations*, allow users to view records for multiple organizations. Organization-visibility rules are applied to screen pops according to the business component data. Verify organization-visibility behavior, changing position manually as needed, for the views involved in your screen pops.

Configuring Communications Login and Logout

Agents can log in to, or log out of, call-center communications systems that work with Siebel Communications Server, either automatically or manually. For more information about login and logout issues, see the following:

- For end user login procedures, see "Using the Communications Toolbar" on page 223.
- For a description of logging in to selected ACD queues and of the end user preference Auto Login to Call Center at Startup (in the Communications options in User Preferences), see "Setting Communications User Preferences" on page 217.
- For descriptions of the configuration parameters AutoLogin, AutoLoginCmd, AutoLogout, AutoLogoutCmd, PreferenceLoginCmd, and PreferenceLogoutCmd, see "Parameters for Communications Configurations" on page 47.

This topic contains the following information:

- "Automatic and Manual Login and Logout" on page 172
- "Login and Logout Command Configuration" on page 173

Automatic and Manual Login and Logout

This topic provides a brief overview of the automatic and manual login and logout capabilities provided for communications-enabled Siebel applications.

Automatic Login

With automatic login, after starting the Siebel application, the end user is automatically logged into all applicable communications systems. Administrators can enable or disable automatic login for all users, or allow each user to specify whether to log in automatically.

When automatic login is in effect, using the Connect command in the File application-level menu in order to log in to a different Siebel database, or in order to log in as a different Siebel user, logs you out of the communications systems and then automatically logs you in again.

Agents who use ACD queues can manually log in to of any of the queues with which they are associated. Agents do this in the Communications options of the User Preferences screen.

Manual Login

With manual login, after starting the Siebel application, the end user clicks the Log In button on the communications toolbar in order to log in to the applicable communications systems. Automatic login must be disabled for all users or for the individual user.

When manual login is in effect, using the Connect command in the File application-level menu in order to log in to a different Siebel database, or in order to log in as a different Siebel user, logs you out of all applicable communications systems and you must log in again manually.

Agents who use ACD queues can manually log in to any of the queues with which they are associated. Agents do this in the Communications options of the User Preferences screen.

Automatic Logout

With automatic logout, when the user exits the Siebel application, the end user is automatically logged out of all applicable communications systems. Administrators can enable or disable automatic logout for all users.

Agents who use ACD queues can manually log out of any of the queues with which they are associated. Agents do this in the Communications options of the User Preferences screen.

Manual Logout

With manual logout, the end user can click the Log Out button on the communications toolbar in order to log out of the applicable communications systems. Automatic logout must be disabled for all users.

Agents who use ACD queues can manually log out of any of the queues with which they are associated. Agents do this in the Communications options of the User Preferences screen.

Login and Logout Command Configuration

This topic documents example command and command data definitions that determine the availability of login and logout functions for the communications toolbar and for menus. For more information:

- About all command and command data parameters shown in this topic, see Chapter 5, “Configuring Events and Commands.”
- About configuring the communications toolbar and communications menu items, see Chapter 6, “Configuring User Interface Elements.”
- About macros such as @PrimaryQueueList, see “Macros for Parameter Values” on page 152.

Login Commands

In the following login commands, SignOnGroup and SignOnGroupInMenu both specify the same set of subcommands, which control login functionality in the communications toolbar and in the Communications submenu of the Tools application-level menu, respectively.

```
[Command:SignOnGroup]
    Hidden          = "True"
    Description     = "Log in"
    ExecuteAll      = "True"
    SubCommand_1    = "LoginToPBX"

[Command:SignOnGroupInMenu]
    MenuPosition    = "20"
    Title           = "Log In"
    Description     = "Log in"
    SubCommand_1    = "LoginToPBX"

[Command:LoginToPBX]
    MenuPosition     = "20.1"
    Title            = "Log In (Phone)"
    DeviceCommand    = "LogIn"
    CmdData          = "LoginToPBX"

    [CmdData:LoginToPBX]
    Param.ACDQueue   = "{@PrimaryQueueList}"
    Param.AgentId    = "{@AgentId}"
    Param.AgentPin   = "{@AgentPin}"
```

Logout Commands

In the following logout commands, SignOffGroup and SignOffGroupInMenu both specify the same set of subcommands, which control logout functionality in the communications toolbar and in the Communications submenu of the Tools application-level menu, respectively.

```
[Command:SignOffGroup]
    Hidden          = "True"
    Description     = "Log out"
    ExecuteAll      = "True"
    SubCommand_1    = "LogoutFromPBX"

[Command:SignOffGroupInMenu]
    MenuPosition    = "22"
    Title           = "Log Out"
    Description     = "Log Out"
    SubCommand_1    = "LogoutFromPBX"

[Command:LogoutFromPBX]
    MenuPosition     = "22.1"
    Title            = "Log Out (Phone)"
    DeviceCommand    = "LogOut"
    CmdData          = "LogoutFromPBX"

    [CmdData:LogoutFromPBX]
    Param.ACDQueue   = "{@PrimaryQueueList}"
```

Configuring Remote Transfers and Conferences

You can configure Siebel Communications Server to allow agents to initiate, transfer, and conference voice calls between call centers, with attached data to support screen pops. This capability might be supported for your CTI middleware.

This topic contains the following information:

- "Creating Communications Configurations" on page 175
- "Specifying Dialing Filters" on page 175
- "Using Macros to Identify Remote Call Centers" on page 176

Creating Communications Configurations

You must create separate communications configurations for each call center. For example, for call centers in San Mateo and Emeryville, create a configuration called San Mateo and a configuration called Emeryville.

Each configuration must define the agents applicable to that physical call center. All configurations must exist in the Siebel database at all call centers. In the example, both the San Mateo and Emeryville configurations exist at both call centers.

NOTE: Session-based communications functionality, such as Siebel CTI, is supported only for single-database environments. Session communications capability is not supported for users of a regional database, to which data is replicated using Siebel Replication Manager.

Specifying Dialing Filters

Each communications configuration must contain configuration parameters specifying dialing filters, in order to support dialing for both locations. These parameters can have names such as DialingFilter.Rule1, DialingFilter.Rule2, and so on. For more information, see "Specifying Parameters for a Communications Configuration" on page 46.

For example, assume that you are configuring communications between two call centers located in the U.S. The San Mateo call center is located in area code 650 and has exclusive use of the prefix 506. The Oakland call center is located in area code 510 and has exclusive use of the prefix 788. Five-digit dialing is supported between these call centers.

With dialing filters such as the examples shown in Table 28, phone numbers for call transfers and call conferences initiated from Siebel Communications Server from either site to the other site (or internal to one site) are converted from ten-digit numbers, as they are stored in the Siebel database, to four-digit numbers, then prepended with a numeral such as *1* or *2* for five-digit dialing.

Table 28. Dialing Filter Parameter Examples

Parameter Name	Parameter Value
DialingFilter.Rule1	510788->1
DialingFilter.Rule2	650506->2

In these examples, the resulting five-digit numbers match extensions defined (with the same number of digits) in the organization's switches and match the extensions defined in the Administration - Communications screen.

The macro $DialingRuleMethod can help you to implement advanced logic for your dialing filters. For more information, see "Macros for Parameter Values" on page 152.

Using Macros to Identify Remote Call Centers

This topic describes how you can use parameters to obtain a call center configuration name.

In the communications configurations for San Mateo and Emeryville, you add a command parameter like one of the examples in this topic to the command data definition for any applicable commands, such as commands used for initiating a call transfer or conference call to another call center.

The command data parameter Param.RemoteConnectStr can include one of two related macros in its parameter value in order to obtain the configuration name:

- **$RemoteConnectStr.** This macro derives the name of a remote call center's communications configuration, using the agent's extension number as a parameter. Table 29 shows examples:

  Table 29. $RemoteConnectStr Examples

Parameter Name	Parameter Value
Param.RemoteConnectStr	[$RemoteConnectStr(@Phone)]
Param.RemoteConnectStr	{$RemoteConnectStr(Owner Phone:Lookup)}

  For more information, see "Macros for Parameter Values" on page 152.

- **$RemoteConnectStr2.** This macro derives the name of a remote call center's communications configuration, using the agent's employee ID as a parameter.

  For example, where the Employee business object and business component are specified, the parameter definition uses the business component field Id to get the name of the configuration. Table 30 shows an example:

Table 30. $RemoteConnectStr2 Example

Parameter Name	Parameter Value
Param.RemoteConnectStr	[$RemoteConnectStr2(Id)]

Example Command Using $RemoteConnectStr

The example command and command data definitions shown in Table 31 and Table 32, using the $RemoteConnectStr macro, use the Owner Phone field in the Service Request business component to determine which call center has the phone number. In this example, the Param.RemoteConnectStr command parameter gets the extension number for the call recipient by macro-expanding Owner Phone:Lookup. The Siebel application looks up which configuration has this extension and returns the name of the call center with this extension.

Table 31. Command: BlindTransferToSROwner

Parameter Name	Parameter Value
Description	Blind Transfer to Service Request Owner
DeviceCommand	TransferMute
Hidden	True

Table 32 shows a command data definition for this command definition.

Table 32. Command Data: BlindTransferToSROwner

Parameter Name	Parameter Value
BusComp	Service Request
RequiredField.'Owner Phone'	?*
Param.PhoneNumber	{Owner Phone:Lookup}
AttachContext	True
Param.CallNotifyText	Blind transfer from {@UserName} about SR {Id}...
Param.RemoteConnectStr	{$RemoteConnectStr(Owner Phone:Lookup)} (In this example, Owner Phone is a field from the Service Request business component.)

Using the Push Keep Alive Driver for Session Connections

The Push Keep Alive communications driver provides a heartbeat message that is useful in maintaining connections for your communications sessions in some environments. You use this driver with Communications Session Manager, such as when you are using Siebel CTI.

The Push Keep Alive driver helps to solve problems that might be experienced in some environments, where push communications sessions between the Application Object Manager and a user's Web browser are sometimes dropped. A connection might be dropped if messages have not been pushed to an agent for a particular length of time.

The Push Keep Alive driver sends a heartbeat message to the Application Object Manager and to each agent user's browser, at a regular interval. This heartbeat message allows the push connection applicable to each agent's communications toolbar not to be dropped in such cases. The heartbeat interval is specified using a driver parameter.

NOTE: When you use the Push Keep Alive driver, you *must* also set the ChannelCleanupTimer configuration parameter to make sure that abandoned connections for communications sessions are dropped promptly. For more information about the ChannelCleanupTimer configuration parameter, see "Parameters for Communications Configurations" on page 47. For information about how to set this parameter, see "Specifying Parameters for a Communications Configuration" on page 46.

Configuring the Push Keep Alive Driver

The following procedure describes how to configure the Push Keep Alive driver. Descriptions of the driver settings and the default driver parameter values follow.

To use the Push Keep Alive driver

1 Start the Siebel application, such as Siebel Call Center, logging in as the system administrator (for example, as the user SADMIN).

2 Navigate to the Administration - Communications screen, then the Communications Drivers and Profiles view.

3 Click the Profiles tab and enter a new record with a name that identifies the profile with the Push Keep Alive communications driver. For example, enter `Push Keep Alive`.

4 In the Profile Parameter Overrides child applet, create records with values similar to those listed in the following table.

Parameter	Value
Name	Select PushKeepAliveTimer from the drop-down list.
Value	Enter the time interval (in seconds) between the Push Keep Alive driver sending messages. For example, enter `180` to have the driver send a heartbeat message every 180 seconds.

5 Click All Configurations in the application link bar and select your configuration in the displayed list.

6 Click the Profiles screen tab.

The list of profiles currently associated with your configuration appears.

7 Click New.

The Add Communications Profiles dialog box appears. This dialog box displays the list of currently available profiles.

8 Select Push Keep Alive, then click OK.

This associates the Push Keep Alive communications driver with your configuration. It means that a heartbeat message is sent at the interval that you specify (for example, every 180 seconds).

9 Do one of the following:

- If you use the Communications Configuration Manager server component to cache your configuration, then shut down and restart your Siebel Server to apply the changes made in steps Step 1 to Step 8.
- If you do not use the Communications Configuration Manager server component, then log out and log into the application for these changes to take effect.

For more information about the Communications Configuration Manager component, see "Administering Communications Configuration Manager" on page 214.

Push Keep Alive Driver Settings

Table 33 lists the settings for the driver record for the Push Keep Alive driver. These settings appear in the Communications Drivers and Profiles view in the Administration - Communications screen.

CAUTION: The values shown in Table 33 are presented for reference only. Do not modify the predefined values for this driver.

Table 33. Push Keep Alive Driver Settings

Field Name	Value
Name	Push Keep Alive
Channel Type	Push Keep Alive
Inbound	Off
Outbound	Off
Interactive	On
Channel String	Push Keep Alive
Library Name	sscmimed

Push Keep Alive Driver Parameters

Table 34 lists the supported parameters for the Push Keep Alive communications driver.

Table 34. Push Keep Alive Driver Parameters

Name	Required	Default Value	Description
LogDebug	No	False	True or False If True, then data output to the log file is more detailed.
LogFile	No	push_{@UserName}.log	The name of the file where logging information is written. If this parameter is empty, then no log file is generated. Logging data is always appended to this file. Log files are created in the log subdirectory of the Siebel Server installation directory. Optionally, if you want to use a single log file in a multichannel environment, then you can specify the same log file name for all of the interactive drivers you are using.
MaxLogKB	No	128	Specifies the maximum size of the log file, in kilobytes (KB). When the log file is full, it is emptied completely, then logging begins again.

Table 34. Push Keep Alive Driver Parameters

Name	Required	Default Value	Description
PushKeep AliveTimer	Yes	0	Specifies the interval, in seconds, for the heartbeat message sent by the Keep Push Alive driver to the Application Object Manager and to each agent's Web browser. For example, if this parameter is set to 180, then the heartbeat message is sent every 180 seconds. The default value, 0, specifies that no heartbeat message is sent. Typically, this parameter is defined as a parameter override in a driver profile that has been associated with the communications configuration.
ReleaseLog Handle	No	True	Indicates that the log file handle for each agent is released periodically, after logs have been generated. The default setting of True provides better performance. When ReleaseLogHandle is False, however, some log files might not be generated if the number of concurrent users exceeds the file-handle capacity provided by the operating system.

Using Business Services with Siebel Communications Server

This topic describes some ways in which you can use Siebel business services with Siebel Communications Server.

Business services represent encapsulated functionality in the form of methods that Siebel application modules or scripts can call within the Siebel application environment. Many types of Siebel Business Applications functionality, including Communications Server functionality, can be accessed as methods of standard business services. Siebel applications running on all client types support Siebel business services.

This topic describes using business services in two general ways:

- Accessing Siebel Communications Server functionality outside of the standard means offered by the Siebel application user interface. For example, Siebel Workflow invokes Communications Server business services, and Siebel VB and Siebel eScript scripts can invoke Communications Server business services.

- Integrating Siebel Communications Server events and commands with Siebel business services, in particular with custom business services or business services other than those provided for Communications Server.

Siebel Communications Server supports multiple business services that support the different functional areas of interactive session communications, outbound communications, and inbound communications.

This topic contains the following information:

Related Books

Siebel Email Administration Guide

Siebel Business Process Framework: Workflow Guide

Configuring Siebel Business Applications

Configuring Siebel Open UI

Using Siebel Tools

Siebel Developer's Reference

Overview: Siebel Enterprise Application Integration (and other books for Siebel EAI)

Invoking Siebel Communications Server Business Service Methods

Siebel Communications Server includes two business services that support session communications for agents:

- **Communications Client.** Supports the communications user interface features such as the communications toolbar and communications menu commands. This business service aggregates the Communications Session Manager business service. It is not involved with server-based communications functionality.
- **Communications Session Manager.** Provides an interface to session-based communications functionality at the server level. Through Server Request Broker and Server Request Processor, this business service communicates with the Communications Session Manager server component. This business service does not deal with the communications user interface.

NOTE: Because the Communications Session Manager business service is aggregated by the Communications Client business service, you can access all Communications Session Manager business service methods from the Communications Client business service. For configuring communications events or commands, the communications toolbar, or the Application Object Manager, you *must* invoke all methods through the Communications Client business service.

These business services work together and can run on the Application Object Manager supporting instances of the Siebel Web Client. They run together with the channel manager, depending on how you have deployed interactive session communications. For more information, see "Enabling Session Communications" on page 201.

You can invoke methods of Communications Server business services from outside Communications Server. You can invoke a method from an applet, a script, a workflow, or another business service, or invoke it in some other way. Siebel Workflow and scripts for Siebel VB or Siebel eScript can invoke Communications Server business services. For more information about Communications Server business services, methods, and arguments, see Appendix B, "Siebel Communications Server Business Services."

About Using Business Services with Events and Commands

This topic describes scenarios for integrating business services with communications event and commands. These scenarios are applicable to a contact center using Siebel CTI and related modules. These scenarios are as follows:

- Invoking a business service method from an event handler
- Invoking a business service method from an event response

  A business service method can also be invoked from an event log, but this scenario is not described here.
- Invoking a communications command from a business service method
- Invoking a business service method from a communications command

Invoking a Command Through the Business Service Model

You can set up a custom business service that invokes a command from a communications configuration. To set this up, you do the following:

- Create a communications command (and a corresponding command data definition) that is intended to be invoked from outside of Siebel Communications Server, and
- Modify or create an applet, script, or business service to invoke a particular communications command

The following examples are for a make-call command and a business service method that invokes this command.

Table 35 and Table 36 show an example of a command that is called by a custom business service. This example executes a MakeCall device command.

Table 35. Command: MakeCallFromCustomService

Parameter Name	Parameter Value
Description	Make Call from Custom Service
DeviceCommand	MakeCall
Hidden	True

Table 36 shows a command data definition for this command definition.

Table 36. Command Data: MakeCallFromCustomService

Parameter Name	Parameter Value
Param.PhoneNumber	{Call Recipient Phone Number}
Param.DisplayText	{My Display Text}

The values for Call Recipient Phone Number and My Display Text are passed from the custom business service to the command when the command is invoked.

A business service must be implemented that invokes the command MakeCallFromCustomService. Communications Server retrieves the values of Call Recipient Phone Number and My Display Text from the input arguments of the business service method and assigns these values to the command data parameters Param.PhoneNumber and Param.DisplayText. The device command MakeCall is executed, using the values for these parameters.

NOTE: By default, communications commands are invoked from business service methods associated with communications toolbar buttons. For more information, see "About Communications Toolbar Configuration" on page 135.

Invoking a Business Service Method from a Command

You can create or modify a command and a corresponding command data definition in the communications configuration to specify a Siebel business service and method to be invoked. To do this, you do the following:

- Use the command parameter ServiceMethod in a command to specify the name of the business service and method to invoke.
- Use the command parameter ServiceParam in the associated command data definition to specify argument names and values to pass to the business service method.

For more information about using these command parameters, see "Commands" on page 116 and "Command Data" on page 128.

In the example shown in Table 37 on page 185 and Table 38 on page 185, Communications Server invokes the method MyMakeCall of the business service MyMakeCallService and passes arguments to the service method. The value of the ServiceParam.PhoneNumber parameter is from the @Phone macro; the value of the ServiceParam.AgentID parameter is from the @AgentId macro.

At run time, commands that invoke business service methods are determined to be enabled or disabled by invoking from the applicable business service the method CanInvokeMethod("*method_name*"), where *method_name* is your business service method to be invoked. If True is returned, then you can invoke the method. If False is returned, then you cannot invoke the method and the command is disabled. No DeviceCommand parameter is included in this example, because the communications command is not designed to call a function on the external communications system.

Table 37. Command: MakeCallInService

Parameter Name	Parameter Value
Description	Make Call In My Service
ServiceMethod	MyMakeCallService.MyMakeCall
Hidden	True

Table 38 shows a command data definition for this command definition.

Table 38. Command Data: MakeCallInService

Parameter Name	Parameter Value
ServiceParam.PhoneNumber	{@Phone}
ServiceParam.AgentID	{@AgentId}

Communications Server can invoke the methods of any business service, not only those, like this example, that pertain to communications.

NOTE: By default, several communications commands invoke business service methods. For examples, see the communications configuration data provided with Siebel Business Applications.

Invoking a Business Service Method from an Event Handler

You can create or modify an event handler in the communications configuration to specify a Siebel business service and method to be invoked. The business service method returns data to determine whether this event handler matches. In addition, it can perform other functions that the customer might want.

To configure this, you use the event parameters ServiceMethod and ServiceParam to specify the name of the business service and method to invoke and to specify argument names and values to pass to the business service method. For more information about using these event parameters, see "Event Handlers" on page 93.

In the example shown in Table 39, Communications Server invokes the method MyMethod of the business service MyService and passes arguments *attribute1* and *attribute2* to the method, using the values of the ServiceParam.Param1 and ServiceParam.Param2 parameters.

Table 39. Event Handler: OnInboundCallReceived

Parameter Name	Parameter Value
ServiceMethod	MyService.MyMethod
ServiceParam.Param1	{attribute1}
ServiceParam.Param2	{attribute1}

The example event handler has the Device Event field set to the appropriate device event for the driver, such as an InboundCall event.

You can use the ServiceMethod and ServiceParam parameters to supplement or replace filter mechanisms using the Filter or FilterSpec parameters. You can also use this method to execute custom code before executing a specified event response. The business method must set a parameter of Result to a value 1 or 0 (or True or False):

- 1 (True) indicates that this event handler matches and is executed, and the associated event response is executed.
- 0 (False) invalidates this event handler; the next event handler is then evaluated.

Invoking a Business Service Method from an Event Response

You can create or modify an event response in the communications configuration to specify a Siebel business service and method to be invoked whenever this event response is invoked by an event handler.

To configure this, you use the event parameters ServiceMethod and ServiceParam to specify the name of the business service and method to invoke and to specify argument names and values to pass to the business service method. For more information about using these event parameters, see "Event Responses" on page 100.

The business service method might also alter the event data fields that were associated with the work item, such as to add new fields.

In the example event response shown in Table 40, the method MyMethod of the business service MyService is invoked to handle the event. The value of the ServiceParam.CallingDN parameter is from the ANI event data field; the value of the ServiceParam.Connection parameter is from the refId event data field.

Table 40. Event Response: OnInboundCallReceived

Parameter Name	Parameter Value
ServiceMethod	MyService.MyMethod
ServiceParam.CallingDN	{ANI}
ServiceParam.Connection	{refId}

When a business service method is invoked from an event response, the value for the output argument Continue is either True (1) or False (0). If Continue is True, then event handling as specified in the event handler that calls this event response proceeds (such as to generate a screen pop).

Integrating with Siebel Scripting Languages

You can use Siebel VB and Siebel eScript scripts in many ways to extend your communications-enabled Siebel applications. You can use Siebel VB and Siebel eScript scripts in the following ways:

- Communications events or commands can invoke Siebel VB or Siebel eScript scripts:
  - Communications events can invoke Siebel VB or Siebel eScript scripts, for instance on the arrival of an incoming work item or upon releasing a work item.
  - Communications commands can invoke Siebel VB or Siebel eScript scripts.
- Siebel VB or Siebel eScript scripts can invoke communications commands (using business service methods) and access event data fields or work item attributes:
  - Siebel VB or Siebel eScript scripts can invoke communications commands. Siebel VB-based menu items, buttons, or toolbar buttons can invoke commands that perform actions such as transferring a work item.
  - Siebel VB or Siebel eScript scripts can access data attached to a work item, such as event data fields like ANI, DNIS, digits collected from an IVR system, or Siebel work item attributes.
  - Siebel VB or Siebel eScript scripts can access methods and arguments of Communications Server business services. For more information, see "Using Business Services with Siebel Communications Server" on page 181 and Appendix B, "Siebel Communications Server Business Services."
  - Siebel VB or Siebel eScript scripts can access run-time data provided by the macros described in "Macros for Parameter Values" on page 152.

Siebel Communications Server integration with Siebel VB and Siebel eScript allows customers to write event handler functions and advanced communications commands in Siebel VB or Siebel eScript.

The methods of all Communications Server business server methods are exposed and can be called with any applicable arguments. All event data fields (such as ANI, DNIS, and so on) are accessible. A Siebel VB or Siebel eScript script can be invoked on any communications event.

A major use of scripting is to allow communications event handling (such as screen-pop logic). But you can also use it to take complete control of the communications technology: to obtain information about the current work item, to intercept events and perform the necessary handling, or to control communications work items (initiate work items, transfer work items, and so on).

NOTE: You can define scripts in the general section of the application object or in a custom business service. Compile them into the Siebel repository (.srf) file for the Siebel Application Object Manager.

This topic contains the following information:

- "Integrating Scripting Using Server and Browser Scripts" on page 188
- "Integrating Scripting Using Server Scripts" on page 191

Related Books

Siebel Object Interfaces Reference

Siebel eScript Language Reference

Siebel VB Language Reference

Configuring Siebel Business Applications

Configuring Siebel Open UI

Siebel Developer's Reference

Using Siebel Tools

Siebel Installation Guide for the operating system you are using

Siebel System Administration Guide

Integrating Scripting Using Server and Browser Scripts

Siebel eScript scripts are generally supported only for the Object Manager layer, not for the browser layer. For Siebel Web Client, the Object Manager is on the Siebel Server (Application Object Manager, such as SCCObjMgr_enu for Siebel Call Center in an ENU environment).

However, Siebel eScript scripts defined on the browser layer can use both server scripts and the @InvokeSWECommand special command. See the example in this topic for creating a communications command, a command for the Siebel Web Engine (SWE), and related server and browser scripts.

NOTE: Transferring data from a server script to a browser script is not supported, due to limitations of SWE commands.

For more information about the @InvokeSWECommand special command, see "Special Commands for Device Commands" on page 87.

Command Example

Table 41 and Table 42 show an example command and associated command data definition that invokes the AlertTest SWE command.

Table 41. Command: Alert

Parameter Name	Parameter Value
DeviceCommand	@InvokeSWECommand
CmdData	Alert

Table 42 shows a command data definition for this command definition.

Table 42. Command Data: Alert

Parameter Name	Parameter Value
Param.SWECommand	AlertTest

SWE Command Example

In Siebel Tools, create a command named AlertTest with the properties shown in Table 43. This SWE command is executed by the communications command shown earlier and in turn invokes the CTI_Test.Alert business service method.

Table 43. SWE Command Example

Property Name	Value
Name	AlertTest
Business Service	CTI_Test
Force Enable	Y
Method	Alert
Target	Browser

Server and Browser Script Examples

Define the following *server script* in the CTI_Test business service. The PreCanInvokeMethod server script in the CTI_Test business service allows the SWE command to be enabled that in turn invokes the browser-layer scripting business service method Alert:

```
function Service_PreCanInvokeMethod (MethodName, &CanInvoke)
{
   var ReturnVal = ContinueOperation;
   if (MethodName == "Alert")
   {
      CanInvoke = "True";
      ReturnVal = CancelOperation;
   }
   return ReturnVal;
}
```

Define the following *browser scripts* in the CTI_Test business service. The PreInvokeMethod browser script invokes the Alert method to create an alert box in the browser.

```
function Service_PreCanInvokeMethod (methodName)
{
   if (methodName == "Alert")
      return true;
   else
      return ("ContinueOperation");
}

function Service_PreInvokeMethod (methodName, inputPropSet, outputPropSet)
{
   if (methodName == "Alert")
   {
      alert("CTI browser script test");
      return ("CancelOperation");
   }
   else
      return ("ContinueOperation");
}
```

In the previous steps, you have successfully created a new communications command named *Alert*. You can execute this command, for example, from the Communications submenu of the Tools menu, or from a keyboard shortcut. Another option is to further integrate this command with a server script, as illustrated in the following server script example:

```
function BrowserScriptTest()
{
   // Invoking Browser Script
   var ctibs = TheApplication().GetService("Communications Client");
   var ip = TheApplication().NewPropertySet();
   var op = TheApplication().NewPropertySet();
   ctibs.InvokeMethod("Alert", ip, op);
}
```

Integrating Scripting Using Server Scripts

Here are some examples of using business services and Siebel scripting with your communications configurations. These examples are for server-side scripting using Siebel eScript.

Event Response Example

Table 44 shows an example event response that invokes the server-layer scripting business service method CTI_Test.EvtBSGotoView. This event response retrieves work item data and generates a screen pop to the view Contact List View, with all contacts whose last name starts with the letter S.

Table 44. Event Response: OnInboundCallReceived

Parameter Name	Parameter Value
ServiceMethod	CTI_Test.EvtBSGotoView
ServiceParam.myWorkItemID	{SiebelWorkItemID}
ServiceParam.ANI	{ANI}

Command Example

Table 45 and Table 46 show an example communications command and associated command data that invokes the server-layer scripting business service method CTI_Test.CmdBSGotoView. This command makes an outbound phone call to the number the user specified in the communications toolbar's text input field.

Table 45. Command: MakeCallToPhone

Parameter Name	Parameter Value
ServiceMethod	CTI_Test.CmdBSGotoView
CmdData	MakeCallToPhone
OnEditControl	True

Table 46 shows a command data definition for this command definition.

Table 46. Command Data: MakeCallToPhone

Parameter Name	Parameter Value
ServiceParam.PhoneNumber	{@Phone:PhoneTypeLookup}
RequiredField.@Phone	?*
Param.CallNotifyText	Call from {@UserName}...

Server Script Examples

Define the following server scripts in the CTI_Test business service. These scripts are invoked by the event response and command described in "Event Response Example" and "Command Example":

```
function Service_PreCanInvokeMethod (MethodName, &CanInvoke)
{
   var ReturnVal = ContinueOperation;
   switch (MethodName)
   {
      case "EvtBSGotoView":
      case "CmdBSGotoView":
      CanInvoke = "True";
      ReturnVal = CancelOperation;
   }
   return (ReturnVal);
}

function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
   var ReturnVal = ContinueOperation;
   switch (MethodName)
   {
      case "EvtBSGotoView":
         EvtBSGotoView(Inputs, Outputs);
         ReturnVal = CancelOperation;
         break;

      case "CmdBSGotoView":
          CmdBSGotoView(Inputs, Outputs);
          ReturnVal = CancelOperation;
          break;
   }
   return (ReturnVal);
}

function EvtBSGotoView(Inputs, Outputs)
{
   // Getting input arguments from Event Response parameters
   var itemID = Inputs.GetProperty("myWorkItemID");
   var ANI = Inputs.GetProperty("ANI");

   // Invoking Business Service "Communications Session Manager" method
   var ctibs = TheApplication().GetService("Communications Session Manager");
   var ip = TheApplication().NewPropertySet();
   var op = TheApplication().NewPropertySet();
   ip.SetProperty("WorkItemID", itemID);
   ctibs.InvokeMethod("GetWorkItemInfo", ip, op);

   // Generate a screen pop to "Contact List View" with
   // all the contacts whose last name start with letter "S"
   var boGlobal = TheApplication().GetBusObject("Contact");
   var bcContact = boGlobal.GetBusComp("Contact");
   bcContact.SetViewMode(AllView);
   bcContact.ClearToQuery();
```

```
      bcContact.SetSearchSpec("Last Name", "S*");
      bcContact.ExecuteQuery(ForwardBackward);
      TheApplication().GotoView("Contact List View", boGlobal);
   }

   function CmdBSGotoView(Inputs, Outputs)
   {
      // Getting input arguments from Command Data parameters
      var number = Inputs.GetProperty("PhoneNumber");

      // Invoking Business Service "Communications Client" method
      var ctibs = TheApplication().GetService("Communications Client");
      var ip = TheApplication().NewPropertySet();
      var op = TheApplication().NewPropertySet();
      ip.SetProperty("PhoneNumber", number);
      ip.SetProperty("ProfileName", "Siebel CTI");
      ctibs.InvokeMethod("MakeCall", ip, op);
   }
```

Integrating with Siebel SmartScript

You can integrate Siebel Communications Server with Siebel SmartScript. For more information about SmartScript, see *Siebel SmartScript Administration Guide*.

This topic contains the following information:

- “Invoking Siebel SmartScript Through Siebel CTI” on page 193
- “Displaying Communications Parameter Data in Siebel SmartScript” on page 194

Invoking Siebel SmartScript Through Siebel CTI

Siebel Communications Server can invoke Siebel SmartScript in order to execute SmartScripts. To enable this behavior, you define parameters for the appropriate event response, using the views in the Administration - Communications screen. If the event response is executed in response to a corresponding event handler matching the call data parameters, then the script is invoked.

The following parameter and value must be added to the event response in order to enable the link between Communications Server and Siebel SmartScript:

- Parameter name: SmartScript.ScriptName
- Parameter value: *Your_script_name*

Any parameters received that are prefixed with *SmartScript* are passed to the SmartScript that is invoked. The available parameters are:

- SmartScript.ScriptName
- SmartScript.ScriptId
- SmartScript.LanguageCode

- SmartScript.CampaignId
- SmartScript.CampContactId
- SmartScript.ContactId

Either ScriptName or ScriptId must be specified in order to start a specific SmartScript. Otherwise, the agent is prompted, through the Choose Script dialog box, for the name of the script to run. In addition, either Siebel VB or Siebel eScript must be invoked to set the correct focus for the applicable business component.

For more information about defining event response parameters, see "Event Responses" on page 100.

Example Events to Invoke Siebel SmartScript Script

Table 47 shows an example event response that invokes a Siebel SmartScript script. The event handler that invokes this event response has the Device Event field set to the appropriate device event for the driver.

Table 47. Event Response: OnInboundCallReceived

Parameter Name	Parameter Value
SmartScript.ScriptName	Customer Service

Displaying Communications Parameter Data in Siebel SmartScript

If a SmartScript is invoked from Communications Server, then call data parameters are also available to be passed to SmartScript. You can access these parameters, which have the same names in SmartScript as in the communications configuration data, through either Siebel VB or Siebel eScript by using the GetParameter function against the SmartScript object. You can include parameters among the variables displayed in the customer dashboard by using the prefix CTI. For example, a variable using the GetParameter function might be defined as follows:

```
var s = GetParameter("CTI.ANI")
```

For more information about variables and programming for SmartScript, see *Siebel SmartScript Administration Guide*.

Integrating with the Customer Dashboard

The customer dashboard, located near the top of the application window for Siebel Call Center and other applications, contains fields with customer-related data for an agent to view. Although the customer dashboard typically displays contact information, you can configure it in Siebel Tools to display other fields.

For more information about configuring the customer dashboard using Siebel Tools, see *Configuring Siebel Business Applications*.

You can configure customer dashboard fields to be populated, or cleared, automatically upon the execution of a communications event or command. For example, the fields might be populated from an event log when an inbound work item arrives, or cleared from an event log if the caller is not found in the database.

The fields in the dashboard are populated with customer data when you invoke a business service and method and pass values from business component fields or event data fields, such as the ANI phone number of a caller. If a match is produced, then the dashboard fields can be populated.

Example Events for Updating and Clearing Customer Dashboard

For example, the following sequence of event handler, event response, and event logs causes the dashboard to be populated with data, or cleared. In these examples, when a single matching contact record is found, the dashboard is populated. When no contact records are found, or multiple records are found, the dashboard is cleared. Adjust your event definitions as necessary, such as if you are using a different business component than Contact.

```
[EventHandler:InboundCallReceived]
    Filter.ANI = "*"
    Profile = ""
    Comments = ""
    Order = "1"
    Response = "OnInboundCallReceived"
    DeviceEvent = "TpAnswered"

[EventResponse:OnInboundCallReceived]
    MultiView = "All Contacts across Organizations"
    QueryBusComp = "Contact"
    QueryBusObj = "Contact"
    QuerySpec = "'Work Phone #'='650506{ANI}'"
    SingleView = "Service Contact Detail View"
    SingleLog = "LogIncomingCallContactFound"
    Log = "LogIncomingCallContactNotFound"
    MultiLog = "LogIncomingCallMultiContactFound"
    Comments = ""
```

If contact data is found, then the following event log updates the customer dashboard:

```
[EventLog:LogIncomingCallContactFound]
    AfterWork.'ACD Call Duration' = "{@WorkDuration}"
    AfterWork.'Done' = "{@Now}"
    AfterWork.'Planned Completion' = "{@Now}"
    BusComp = "Action"
    BusObj = "Contact"
    Display = "False"
    LogField.'Account Id' = "{Contact.'Account Id'}"
    LogField.'Call Id' = "{refId}"
    LogField.'Contact Id (Thin)' = "{Contact.Id}"
    LogField.'Planned' = "{@WorkStartTime}"
    LogField.'Started' = "{@WorkStartTime}"
    LogField.Description = "Inbound call"
```

```
        LogField.Type = "Call - Inbound"
        ServiceMethod = "Persistent Customer Dashboard.Update Dashboard from CTI"
        ServiceParam.Field = "Id"
        ServiceParam.Value = "{Contact.Id}"
        WorkTrackingObj.ContactId = "{Contact.Id}"
        Comments = ""
```

If no contact record is found, then the following event log clears the customer dashboard:

```
    [EventLog:LogIncomingCallContactNotFound]
        AfterWork.'ACD Call Duration' = "{@WorkDuration}"
        BusComp = "Action"
        BusObj = "Contact"
        LogField.'Call Id' = "{refId}"
        LogField.Description = "Unknown Caller({ANI})"
        LogField.Type = "Call - Inbound"
        ServiceMethod = "Persistent Customer Dashboard.CleanDashBoard_UI"
        Comments = ""
```

If multiple contact records are found, then the following event log clears the customer dashboard:

```
    [EventLog:LogIncomingCallMultiContactFound]
        AfterWork.'ACD Call Duration' = "{@WorkDuration}"
        BusComp = "Action"
        BusObj = "Contact"
        LogField.'Call Id' = "{refId}"
        LogField.Description = "Inbound call({ANI})"
        LogField.Type = "Call - Inbound"
        ServiceMethod = "Persistent Customer Dashboard.CleanDashBoard_UI"
        Comments = ""
```

Generating Contact Center Telephony Analytics Reports

The Contact Center Telephony Analytics screen provides reports that allow organizations to analyze all aspects of their contact center performance. Contact center administrators can generate reports using the following views:

- **Contact Center and Agent Performance.** Provides detailed insight into agent performance compared to standard targets and benchmarks.
- **Contact Center Sales.** Provides the integration of contact center data with actual detailed sales activity and performance data in order to provide a clear understanding of how the enterprise contact center achieves sales.
- **Customer Service.** Provides overall customer service metrics and an understanding of the typical customer experience through the various channels.
- **Service Delivery and Costs.** Provides the complete view of an initial incident through issue resolution with integrated expense data, allowing a start-to-finish cost analysis for an entire service call, including field service delivery activities and related data.

For more information about Siebel Business Intelligence products that support the reports in the Contact Center Telephony Analytics screen, see *Siebel Reports Guide*.

To generate Contact Center Telephony Analytics reports

1. Navigate to the Contact Center Telephony Analytics screen.
2. Click to select one of the following Contact Center Telephony Analytics views:
   - Contact Center and Agent Performance
   - Contact Center Sales
   - Customer Service
   - Service Delivery and Costs

   The selected view displays reports for you to view or print.

Viewing Communications Status Data

System administrators and call-center administrators can display status data about agents who are engaged in communications activity in Siebel applications. Status data is displayed in:

- All Active Agent Status view
- All Channel Items view

This topic contains the following information:

- "Viewing Agent Status Data" on page 197
- "Viewing Channel Status Data" on page 198

Viewing Agent Status Data

You can use the All Active Agent Status view to view status data for each agent who is currently logged in and using a Siebel application configured to support session communications. For each agent, the view displays:

- The agent's login, agent ID, and first and last name
- The communications configuration that is active for the agent
- The teleset the agent is using, for voice agents
- Currently active work items for the agent, for each channel
- The agent's state (such as Ready, Not Ready) for each channel for the agent's configuration

To view agent status data

1. Navigate to the Administration - Communications screen, then the All Active Agent Status view.

2 Click Refresh to update the displayed data.

NOTE: The All Active Agent Status view is always updated by the Siebel CRM system software and does not require specific parameter settings or event log definitions.

Viewing Channel Status Data

You can use the All Channel Items view to view status data about current communications work items for agents. For each work item, the view displays the:

- Work item's channel, such as voice, email, and so on
- Channel target address
- Time the work item started
- Agent's Siebel login, agent log, and agent password

NOTE: For each work item, the value displayed in the Start Time field is accurate only to the extent that the time is synchronized between the various computers running instances of Communications Session Manager or Application Object Manager.

To view channel status data

1. Navigate to the Administration - Communications screen, then the All Channel Items view.
2. Click Refresh to update the displayed data.

Modifying How Channel Status Data Is Displayed

Channel status data is logged for display in the All Channel Items view only if the UpdateChannelStatusTable configuration parameter is set to True. In communications configurations provided with Siebel Business Applications, this parameter is set to True. For more information, see "Specifying Parameters for a Communications Configuration" on page 46.

NOTE: Records in the All Channel Items view reflect current activity only. If any records persist in this view for past work items, then you might have to delete such records manually in this view.

Distributing Status Data to Agents

Call-center administrators might want to distribute call-center statistics from the switch or CTI middleware to call-center agents for display in the message broadcast area of the Siebel client. For example, it might be useful for an agent to know how long a call has been in an ACD queue before taking the call, how many calls are currently in an ACD queue, or the average talk time for a call. Consult the documentation for your switch or CTI middleware for information about how to retrieve call-center statistics.

You configure message broadcasting in the Message Broadcasts view, located in the Administration - Communications screen. For more information, see *Siebel Applications Administration Guide*.

8 Administering Siebel Communications Server

This chapter describes how to configure and run Siebel Communications Server components, including how to enable session communications for your applications. It includes the following topics:

- Siebel Server Requirements for Siebel Communications Server on page 199
- Enabling Session Communications on page 201
- Administering Communications Session Manager on page 213
- Administering Communications Configuration Manager on page 214

Siebel Server Requirements for Siebel Communications Server

In order for you to be able to use Siebel Communications Server within your enterprise, you must ensure the following requirements for the Siebel Server.

- When you configure the Siebel Server after installation, you must include the Communications Management (CommMgmt) component group.
- You must enable the Communications Management component group.

The topics that follow describe the Siebel Communications Server components and describe additional Siebel Server requirements.

NOTE: For more information about running, configuring, or monitoring Siebel Server components, see *Siebel System Administration Guide*.

This topic contains the following information:

- "Server Components for Siebel Communications Server" on page 199
- "Running Siebel Communications Server in Heterogeneous Server Environments" on page 200
- "Using Siebel Server Load Balancing with Siebel Communications Server" on page 201

Server Components for Siebel Communications Server

The Communications Management component group includes these server components:

- **Communications Session Manager (CommSessionMgr).** Supports multichannel user-interactive sessions for agents using the communications toolbar for voice, email, or other types of work items. For more information, see "Enabling Session Communications" on page 201 and "Administering Communications Session Manager" on page 213.

- **Communications Configuration Manager (CommConfigMgr).** Loads and caches communications configuration data for agents using functions supported by Communications Session Manager. For more information, see "Administering Communications Configuration Manager" on page 214.
- **Communications Inbound Receiver (CommInboundRcvr).** Receives inbound work items and, in some deployments, queues them for processing by Communications Inbound Processor. Work items can include email messages (for Siebel Email Response). For more information, see *Siebel Email Administration Guide*.
- **Communications Inbound Processor (CommInboundProcessor).** For deployments of Siebel Email Response supporting nonreal-time work items, this component processes inbound work items that were queued by Communications Inbound Receiver. For more information, see *Siebel Email Administration Guide*.
- **Communications Outbound Manager (CommOutboundMgr).** Processes outbound communications, for email, fax, wireless messages, or page channels. Supports communication requests, whether directly or through Siebel Workflow. Also supports outbound capabilities for Siebel Email Response and for the Send Email, Send Fax, and Send Wireless Message commands. For more information, see *Siebel Email Administration Guide*.
- **Page Manager (PageMgr).** Used to send pages by the Send Page command. Also used by some workflow processes in Siebel Workflow. For more information about setting up and using Page Manager, see *Siebel Business Process Framework: Workflow Guide*.
- **Email Manager (MailMgr).** Used to send email by some workflow processes in Siebel Workflow. For more information about setting up and using Email Manager, see *Siebel Business Process Framework: Workflow Guide*.

Running Siebel Communications Server in Heterogeneous Server Environments

Siebel Communications Server supports heterogeneous server environments. Because Communications Server components such as Communications Session Manager load driver library files, the applicable components must be enabled on the same platform that the driver was developed for. For heterogeneous operation, the Application Object Manager can run in parallel on a different platform. For more information, see My Oracle Support Certifications, and see also "About Communications Drivers and Profiles" on page 31.

Using Siebel Server Load Balancing with Siebel Communications Server

If you use Siebel Server load balancing to implement load balancing among multiple instances of Siebel Server, then, for each Siebel Server participating in load balancing, you must enable the Communications Management component group, as noted, and enable or start any applicable Communications Server components.

If you are also using Communications Session Manager, then you must set the Enable Communication parameter to True for each Application Object Manager instance. For more information about enabling communications, see "Enabling Session Communications" on page 201.

NOTE: Siebel Server load balancing does not load balance requests for Siebel Communications Server. Rather, in this case the Application Object Manager from which a Communications Server request is made is load-balanced and requires that the necessary Communications Server components be enabled and running.

For more information about installing and configuring Siebel Server and Siebel Server load balancing, see the *Siebel Installation Guide* for the operating system you are using and *Siebel System Administration Guide*.

Enabling Session Communications

Interactive session-based communications can be enabled for employees who use the Siebel Web Client or the Siebel Developer Web Client (the Siebel Mobile Web Client in connected mode).

NOTE: The Siebel Developer Web Client is supported for administration, development, and troubleshooting usage scenarios only.

Session-based Communications Server functionality can be accessed either through server components or business services, depending on your Siebel client deployment choices.

NOTE: This topic applies only to enabling communications activities that use the communications toolbar or menu commands for handling voice and email channels for interactive communications.

For information about using the communications toolbar and menu items, see "Using the Communications Toolbar" on page 223 and "Using Communications Menu Commands" on page 232.

For information about the communications architecture, see "About Siebel Communications Server Architecture" on page 18.

This topic contains the following information:

- "About Communications Session Modes" on page 202
- "Prerequisites for Enabling Session Communications" on page 202
- "Parameters for Application Object Manager and Siebel Developer Web Client" on page 203
- "Parameters for Communications Session Manager" on page 211
- "Enabling Communications Sessions for Siebel Web Client" on page 212
- "Enabling Communications Sessions for Siebel Developer Web Client" on page 212

About Communications Session Modes

Supported modes of operation for session communications are as follows:

- **Siebel Web Client.** Server-based communications session. (Most customers are likely to use this mode.)
- **Siebel Developer Web Client.** Two modes are available:
  - Server-based communications session
  - Local communications session

Server-based communications sessions use the Communications Session Manager server component. Local communications sessions do not use this component, but rather perform communications processing locally.

The Communications Client and Communications Session Manager business services run on the Application Object Manager for Siebel Web Client deployments. These business services run locally for Siebel Developer Web Client deployments.

To enable communications, you configure the Siebel Web Client through the Application Object Manager. You configure Siebel Developer Web Client through the application configuration file, such as uagent.cfg for Siebel Call Center.

Prerequisites for Enabling Session Communications

In order for agents to be able to use communications features, you must set up your communications environment. For more information, see "Process of Configuring Siebel CTI" on page 24. In particular:

- Siebel communications-related products (such as Siebel CTI, or Siebel Email Response) must be licensed, configured, and available. Applicable third-party communications systems also must be configured and available.
- For scenarios requiring the Communications Session Manager or Communications Configuration Manager server component, these Siebel Server components must be enabled and running. Also, they must be available for the instances of the Application Object Manager that connect to them for each agent's communications session.
- Call center users must have been defined as agents within a communications configuration, as described in "Specifying Agents" on page 55. (Demo users are predefined in the sample communications configuration data.)
- For CTI (voice-call handling), telesets and extensions must be defined, and the telesets must be associated with actual users who run the Siebel application.

Parameters for Application Object Manager and Siebel Developer Web Client

This topic describes the parameters for the Application Object Manager (for Siebel Web Client) and Siebel Developer Web Client. These parameters are described in Table 48 on page 204.

NOTE: The Siebel Developer Web Client is supported for administration, development, and troubleshooting usage scenarios only.

The type of client that you use determines how the parameters are specified:

- For deployments of the Siebel Web Client, parameters are specified as server component parameters for the Application Object Manager that is to support communications users.
- For deployments of the Siebel Developer Web Client (where applicable), parameters are specified in the [Communication] section of the application configuration file, such as uagent.cfg for Siebel Call Center, for each communications user's Siebel client.

NOTE: For the parameters listed in Table 48 on page 204, if the parameter value is changed while the Application Object Manager is running, then the changed value affects communications sessions for all users who log in subsequently.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommConfigCache	Communication Configuration Cache	False	Enables or disables the caching of communications configuration data on the Application Object Manager. **NOTE:** This parameter applies to the Siebel Web Client only, not to the Siebel Developer Web Client. ■ When this parameter is False, communications configuration data is loaded separately for each agent session, with no caching on the Object Manager. ■ When this parameter is True, the Application Object Manager loads and caches communications configuration data. When CommConfigCache is True, after the first agent logs in, subsequent agent logins for the same configuration download the configuration data from the cache. This speeds up the login process and reduces memory usage for each agent session. This parameter can be used with or without the Communications Configuration Manager component and the CommConfigManager parameter. For more information, see the description of the CommConfigManager parameter.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommConfigManager	Communication Configuration Manager	False	Enables or disables use of the Communications Configuration Manager (CommConfigMgr) for loading and caching communications configuration data to reduce configuration downloads. ■ When this parameter is False, communications configuration data is loaded separately for each agent session, with no caching. ■ When this parameter is True, Communications Configuration Manager is employed to load and cache communications configuration data. After the first agent logs in, subsequent agents using the same configuration download the configuration data from the cache, thereby speeding up the login process. For more information, see "Administering Communications Configuration Manager" on page 214.
CommConfigManager Name	Communication Configuration Manager Name	CommConfigMgr	Specifies the name of the Communications Configuration Manager server component to use when the parameter CommConfigManager is True. A global deployment might, for example, use different components for different languages within the same enterprise.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommEnable	Enable Communication	False	Enables or disables session-based communications for agents. ■ When this parameter is True, the communications toolbar and menu elements are displayed and activated for agents who are associated with a configuration. ■ When this parameter is False, the communications toolbar and menu elements are not displayed. If an agent is not associated with a configuration, then the communications toolbar is not displayed, even if CommEnable is True. Enabling communications has many additional requirements, which are described in Chapter 4, "Configuring Siebel CTI."

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommLocalDriver	Local Communication Driver	False (as component parameter) True (as configuration file parameter for Siebel Developer Web Client)	Specifies on which computer interactive communications drivers are loaded by the channel manager for each agent communications session: ■ For either the Siebel Web Client or the Siebel Developer Web Client: when this parameter is False, interactive drivers are loaded on the Siebel Server computer where Communications Session Manager is running. ■ For the Siebel Developer Web Client: if this parameter is True in the configuration file, then interactive drivers are loaded on each agent's Siebel client. ■ For the Siebel Web Client: if this parameter is True on the Application Object Manager, then interactive drivers are loaded on the Application Object Manager. For more information, see "Administering Communications Session Manager" on page 213.
CommLogDebug	Log Debug Message	False	Specifies whether the log file specified with the CommLogFile parameter must contain extra detail.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommLogFile	Log File	SComm.log	Specifies the name of the log file to create for each agent's communications session. For example, where the parameter is set to SComm.log, files named SComm_*user*.log are created, where *user* is each agent's Siebel login name. These files are written in the `log` subdirectory of the Siebel Server or Siebel Developer Web Client installation directory.
CommMaxLogKB	Maximum Log Size (KB)	1024	Specifies the maximum size, in Kilobytes, of log files specified using CommLogFile.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommMaxMsgQ	Maximum Message Queue	64	The maximum number of messages to be queued for the agent's communications toolbar. If the number of messages exceeds the maximum, then the oldest message is discarded. For example, with the default value of 64, the most recent 64 messages are queued. If a 65th message arrives, then the oldest message in the queue is discarded in order for the new message to be received. It is recommended that you use the default setting unless you require changing it. If you set this value too low, then older messages might be deleted in some cases, such as in a slow network environment. If you set this value too high, then server resources might be overused, by storing many messages for agents who experience browser problems that interrupt the session. The behavior is also subject to the general session timeout defined for Siebel Web Client deployments. If an agent's session times out, then all toolbar messages for the session are cleared.

Table 48. Object Manager and Developer Web Client Parameters for Communications Sessions

Parameter Name	Display Name	Default Value	Description
CommReleaseLogHandle	Release Log Handle	True	True indicates that the log file handle for each agent is released after logs have been generated. The default setting of True provides better scalability because the handles are released, but it can slow down performance in some cases. When CommReleaseLogHandle is True, some log files might not be generated if the number of concurrent users exceeds the file-handle capacity provided by the operating system.
CommReqTimeout	Request Timeout	600	Specifies the number of seconds to wait for a response from the Communications Session Manager for a particular communications interaction.

Parameters for Communications Session Manager

Parameters relating to log files can be specified as server component parameters for Communications Session Manager. These parameters are described in Table 49 on page 211.

NOTE: For the parameters listed in Table 49 on page 211, if the parameter value is changed while the Communications Session Manager is running, then the changed value affects communications sessions for all users who log in subsequently.

Table 49. Communications Session Manager Parameters

Parameter Name	Display Name	Default Value	Description
LogDebug	Log Debug Message	False	Specifies whether the log file specified with the LogFile parameter must contain extra detail.
LogFile	Log File	SComm.log	Specifies the name of the log file to create for each agent's communications session. For example, where the parameter is set to SComm.log, files named SComm_*user*.log are created, where *user* is each agent's Siebel login name. These files are written in the log subdirectory of the Siebel Server installation directory.
MaxLogKB	Maximum Log Size (KB)	1024	Specifies the maximum size, in Kilobytes, of log files specified using LogFile.
ReleaseLogHandle	Release Log Handle	True	Indicates that the log file handle for each agent is released periodically, after logs have been generated. The default setting of True provides better performance. When ReleaseLogHandle is False, some log files might not be generated if the number of concurrent users exceeds the file-handle capacity provided by the operating system.

Enabling Communications Sessions for Siebel Web Client

End users can access communications using the Siebel Web Client by connecting the Web browser to an Applications Object Manager for which server component parameter values are set as described in this topic. For more information about setting server component parameter values, see *Siebel System Administration Guide*.

To enable or disable session communications for Siebel Web Client users

- On each Application Object Manager that is to support users for whom session communications must be available, set the server component parameter CommEnable (Enable Communication) to True. Alternatively, to disable communications, set it to False. To set this parameter, do one of the following:
  - Navigate to the Administration - Server Configuration screen, then the Servers view. Specify the Siebel Server, and select Call Center Object Manager (for example) in the Components list. Set a value for the Enable Communication parameter in the Parameters list (for this component).
  - On the command line, run Siebel Server Manager, specifying your Siebel Gateway Name Server, Siebel Enterprise Server, and Siebel Server, as appropriate. In Server Manager, enter (on one line) a command like the following, as appropriate for your Application Object Manager component:

    ```
    change parameter CommEnable=true for component SCCObjMgr_enu
    ```

After you set the CommEnable parameter value, the change takes effect for each subsequent user login. Optionally, you can ask existing users to log out and log in again. For more information about using Siebel Server Manager (GUI or command line), see *Siebel System Administration Guide*.

Enabling Communications Sessions for Siebel Developer Web Client

This topic describes how you can enable communications using the Siebel Developer Web Client, which is the Siebel Mobile Web Client in connected mode. Enabling communications is subject to your having the appropriate license keys. Oracle does not support the Siebel Developer Web Client in production environments. It is recommended that you restrict the use of this client to development and test environments. Support for this client in these environments is restricted.

Enabling Communications Using Command-Line Option

You can enable communications in the Siebel Developer Web Client by including the command-line option /ct at the end of the command line, such as in the application shortcut. Any of the Siebel Business Applications can support this command-line option.

Enabling Communications by Modifying Configuration File

You can enable communications in the Siebel Developer Web Client by setting the following parameter in the [Communication] section of the application configuration file, such as uagent.cfg for Siebel Call Center:

```
CommEnable = True
```

Administering Communications Session Manager

This topic describes how to administer the server component Communications Session Manager. The short name for this component is CommSessionMgr. This server component uses generic configuration parameters and does not have to be configured. However, you can configure logging parameters. For more information, see "Parameters for Communications Session Manager" on page 211.

Overview of Communications Session Manager

The Communications Session Manager supports multichannel user-interactive sessions for agents using the communications toolbar for voice, email, or other types of work items. It manages agent sessions, for which applicable interactive communications drivers are loaded into memory.

For most Siebel client deployment choices, the Communications Session Manager must be available for the instances of the Application Object Manager that connect to it for each agent's communications session.

In each case, the component that is employed is identified through the applicable data source for the Siebel application session. For each data source, a specific Siebel Gateway Name Server, Siebel Enterprise Server, and Siebel Server are associated. The Communications Session Manager component must be enabled and running on this Siebel Server.

Alternatively, you can specify the following communications configuration parameters, to identify the Communications Session Manager to use:

- **GatewayAddress.** Specify the Siebel Gateway Name Server. For example, specify a value like *gateway-host*.
- **EnterpriseServer.** Specify the Siebel Enterprise Server. For example, specify a value like *siebel*.
- **RequestServer.** Specify the Siebel Server. For example, specify a value like *server-host*.
- **CommSessionMgr.** Specify the Communications Session Manager. For example, specify a value like *CommSessionMgr*.

If an agent logs in using a configuration containing these parameters, then the agent is connected to the component identified by these values. For more information about these parameters, see "Specifying Parameters for a Communications Configuration" on page 46.

For information about how the CommLocalDriver parameter relates to the Communications Session Manager component, see "Parameters for Application Object Manager and Siebel Developer Web Client" on page 203.

When Communications Session Manager Is Unavailable

If the Communications Session Manager stops running or becomes unavailable for some reason, then all connected users (agents) receive the following message in an alert-type dialog box in their browser: *Connection to Communications Server is down. Toolbar is reset!*

For each such agent, the communications toolbar displays as if no communications driver is loaded. That is, the toolbar is displayed but most of the toolbar buttons are unavailable.

Running Communications Session Manager

When the Communications Management component group is enabled, the Communications Session Manager component is started automatically. For any computer on which you do not want to run Communications Session Manager, configure the Siebel Server not to start it.

Communications Session Manager is a batch-mode component. It relies on the services of the Server Request Broker and Server Request Processor server components. These components must be running on the Siebel Server for communications to be processed successfully.

NOTE: If your CTI middleware server is restarted, then, depending on the third-party vendor requirements, you might also have to restart the Communications Session Manager server component that connects to it.

For more information about configuring, starting, and stopping Siebel Server components, see *Siebel System Administration Guide*.

Administering Communications Configuration Manager

This topic describes how to administer the server component Communications Configuration Manager. The short name for this component is CommConfigMgr. This server component uses generic configuration parameters and does not have to be configured.

The Communications Configuration Manager enhances application performance when agent communications sessions are initiated. It does this by reducing or eliminating the downloading of communications configuration data for each session. After the first agent login that uses a particular configuration, for all subsequent logins with the same configuration, the configuration is loaded from a cache rather than loaded from the database.

For information about enabling the use of this component using the CommConfigManager parameter, see "Parameters for Application Object Manager and Siebel Developer Web Client" on page 203.

The Communications Configuration Manager must be available for the instances of the Application Object Manager that connect to it for each agent's communications session.

In each case, the component that is employed is identified through the applicable data source for the Siebel application session. For each data source, a specific Siebel Gateway Name Server, Siebel Enterprise Server, and Siebel Server are associated. The Communications Configuration Manager component must be enabled and running on this Siebel Server.

NOTE: Enable the Communications Configuration Manager only after the communications configuration has been thoroughly tested and you are ready to deploy it to your agents. If any changes are made to the communications configuration, then you must restart the Communications Configuration Manager component in order to refresh the cached configuration data.

Running Communications Configuration Manager

When the Communications Management component group is enabled, the Communications Configuration Manager component is started automatically. For any computer on which you do not want to run Communications Configuration Manager, configure the Siebel Server not to start it.

For more information about configuring, starting, and stopping Siebel Server components, see *Siebel System Administration Guide*.

9 Communications Operations for End Users

This chapter provides information about end-user operations relating to communications capabilities. It includes the following topics:

- Setting Communications User Preferences on page 217
- Using the Communications Toolbar on page 223
- Using Communications Menu Commands on page 232

Most of the tasks described in this chapter are typically performed by users such as call center agents who are primary users of communications features. These tasks might also be performed by users such as call center managers or communications administrators.

NOTE: End users are not expected to directly use this guide. Your Siebel application implementation might differ from what is described in this chapter. This chapter contains references to administration topics that relate to the end-user operations described here.

Setting Communications User Preferences

The User Preferences screen contains two sets of communications-related settings, Outbound Communications and Communications. Users can modify, or might have to modify, these settings depending on the needs of your business or on the users' preferences. The Communications options also include a list of the agent's ACD queues, which the agent can selectively log in to or log out of.

Preferences are stored separately for each user and stored based on Siebel client type:

- Preference settings specified in the Siebel Web Client are stored on the Siebel Server computer.
- Preference settings specified in the Siebel Mobile Web Client are stored on the user's local computer.

NOTE: It is recommended that end users only change preference settings at the direction of the system administrator or call-center administrator.

This topic contains the following information:

- "Specifying Communications Preferences" on page 218
- "Preference Settings for Communications" on page 218
- "Logging In to or Out of an ACD Queue" on page 222

Specifying Communications Preferences

This topic describes how to specify communications preferences.

To specify communications preferences

1. From the application-level menu, choose Tools, then User Preferences.

   The User Preferences screen is displayed.

2. From the link bar, choose either Outbound Communications or Communications.

   The appropriate set of communications preferences are displayed.

3. Specify the preferences. Preference settings are described in the following topics:

   - "Preference Settings for Communications" on page 218
   - "Logging In to or Out of an ACD Queue" on page 222

4. As necessary for the particular setting, log out of the Siebel application and log in again.

   The descriptions for each user preference setting indicate whether this step is required.

Preference Settings for Communications

This topic describes in detail the preferences in the Communications options of the User Preferences screen. The preference settings are grouped by category. Each preference applies only to certain communications features.

General Preferences

The following are the communications preferences in the General category. These preferences apply to users who are part of a communications configuration. Such users typically handle voice calls using Siebel CTI, or receive inbound work items using Siebel Email Response.

- **Receive Screen Pop.** If a user wants not to be the recipient of screen pops upon communications events, such as receiving a new or transferred work item, then the user can clear this check box to disable receiving screen pops.

  If this option is not checked, then no screen pops are received. By default, each user receives screen pops.

- **Send Screen Pop.** When this check box is checked, attaching Siebel client context data is enabled for work item transfers and conferences and for internal calls placed by the agent.

  If this option is not checked, then context data is not attached, and no screen pops are sent. By default, each communications user sends screen pops.

  For more information about requirements for screen pops, see "Event Responses" on page 100.

- **Bring Siebel to Front.** This drop-down list specifies whether, or when, the Siebel client window is restored from being minimized and moved to the front of the screen on a communications event such as an inbound work item.

  By specifying either On All Incoming Work Items or On Matching Events, the agent can make sure that the Siebel application window is displayed when it is needed so the agent can use the communications toolbar or other features, or view screen pops. Screen pops, if they are enabled, are still generated regardless of the setting of the Bring Siebel to Front option.

  NOTE: The behavior of this feature varies depending on your browser and operating system version. In some cases, a Siebel client window might not move to the front. In this case, restore the window from the taskbar. Bring Siebel to Front does not work in deployments using Siebel Open UI. You can set a parameter value to cause the browser title to change in this case. For more information, see the description of the BringSiebeltoFrontBrowserTitle parameter for communications configurations, in "Parameters for Communications Configurations" on page 47. For more information about Siebel Open UI, see *Configuring Siebel Open UI*.

  The Bring Siebel to Front options are:

  - **On All Incoming Work Items.** Brings the Siebel client window to the front on all communications events received at the agent's computer that represent incoming work items. This option is the default and is best for general use by agents running a single instance of the Siebel client.
  - **On Matching Events.** Brings the Siebel client window to the front on communications events that match that instance of the Siebel client.

    This option must be used only by agents running multiple instances of the Siebel client that are each enabled for session communications, but that have different settings, such as for language or locales. In such a situation, different event handlers might be defined, for example, for each language or locale. Use this setting only at the instruction of your administrator.

    Administrators can find more information by referring to the description of the MaxCommToolbars configuration parameter, in "Parameters for Communications Configurations" on page 47.
  - **Off.** Does not bring the Siebel client window to the front on communications events. The window might remain hidden behind another window or minimized.

- **Auto Login to Call Center at Startup.** When this check box is checked, an agent automatically logs in to the call center's communications systems when starting the Siebel client or connecting to the Siebel database.

  What the agent logs in to depends on how your communications environment has been deployed and how login commands are configured in the communications configuration. Agents can log in to ACD queues. Automatic login can make it easier for your users to begin receiving inbound work items from these sources.

  Voice agents who are authorized users for more than one teleset might be best advised not to use automatic login, so they can verify which extension they are using before logging in.

  NOTE: Generally, end users must change this setting only at the direction of the system administrator or call-center administrator.

Whether this control can be set by agents depends on the setting of the AutoLogin configuration parameter. Administrators can find more information under "Parameters for Communications Configurations" on page 47 and "Configuring Communications Login and Logout" on page 172.

- **Message Display Interval.** Specifies the length of time, in seconds, that a message displays in the status bar above the communications toolbar. (An example message might be *Call from 6505060000.*) The default value is 7 seconds.

  Whether this control can be set by agents depends on the setting of the MessageDisplayInterval configuration parameter. Administrators can find more information under "Parameters for Communications Configurations" on page 47.

Sound Settings

The following are the communications preferences in the Sound category. These preferences apply to users who are part of a communications configuration. Such users typically handle voice calls using Siebel CTI, or receive inbound work items using Siebel Email Response:

- **Enable Sound.** Lets the agent specify if a sound file plays when an inbound work item arrives. By default, no sound file plays.

  If Enable Sound is checked, then when a work item arrives, if no valid sound file is specified, the computer might beep. For more information, see the description of the Sound File user preference.

- **Sound File.** Lets the agent specify which sound file to play when the Enable Sound user preference is checked. For example, you can specify ringin.au, ringin.wav, or another such file that is valid for your environment. For more information, see the description of the Enable Sound user preference.

  The default sound file is shown in the communications preferences as `files/ringin.au`. The sample sound files ringin.au, ringin.wav, and ringin.mp3 are installed in `webmaster\files` (Microsoft Windows) or `webmaster/files` (UNIX), in the Siebel Server installation directory.

  The default value for the Sound File user preference includes a relative URL corresponding to the `files` directory on the Siebel Web Server Extension (SWSE) installation directory. Files from the Siebel Server location mentioned earlier are copied to this location.

  NOTE: Each time the Web server is restarted, files and subdirectories are automatically copied from the `webmaster` directory on the Siebel Server to the `public` directory on the SWSE. You can also update files manually without restarting the Web server. For more information, see the *Siebel Security Guide* and the *Siebel Installation Guide* for the operating system you are using.

  Alternatively, you can specify the full path for a sound file that is to be loaded, if the file is located in a different directory *and* if (for the high-interactivity client) the Java applet for the communications toolbar is signed. For example, the following might be valid paths for specifying a sound file ringin.wav located on a user's local computer or on a network location:

  - `e:\dev\ringin.wav` (for Microsoft Windows)
  - `\\host\dev\ringin.wav` (for Microsoft Windows)
  - `file://e:\dev\ringin.wav` (for Microsoft Windows)
  - `http://`*`network_path`*`/ringin.wav` (for Microsoft Windows or UNIX)

- `/usr/ringin.wav` (for UNIX)

For deployments using Siebel Open UI, the following limitations apply:

- The Sound File user preference does not support specifying a local path for a sound file.
- If sound is enabled but no sound file is specified, then the browser does not beep.
- The browser must support HTML 5 in order to play a sound file.
- Whether a particular type of sound file can play depends on the browser.

For more information about Siebel Open UI, see *Configuring Siebel Open UI*.

Teleset and Configuration Settings

The following are the communications preferences in the Teleset and Configuration category. These preferences apply to users who are part of a communications configuration. Such users typically handle voice calls using Siebel CTI, or handle other types of inbound work items through the communications toolbar.

The first two options described apply only to agents who handle voice calls using Siebel CTI.

- **Teleset.** Lets the agent specify the teleset to use the next time the agent logs in.

  The telesets an agent can choose from are those that have been associated with the agent. These telesets were specified when the teleset, agent, and extension data was entered in the views in the Administration - Communications screen. The Standard Extension field lists extensions for the currently selected teleset.

  An agent can choose a teleset from the Teleset field only if at least one teleset has been associated with the agent, and the agent is not using a hoteling teleset.

  If the agent has logged into a hoteling computer, then the current hoteling teleset is shown in this field; the agent cannot change it.

  NOTE: After you choose a different teleset and extension, you must log out and log in again in order for the new extension to be in effect.

  Administrators can find more information under "Specifying Telesets" on page 58.

- **Standard Extension.** Lets the agent specify the standard extension to use, from the teleset selected in the Teleset field, the next time the agent logs in.

  The available extensions to choose from are those of the current teleset. These extensions were specified when the teleset, agent, and extension data was entered in the views in the Administration - Communications screen. Standard extensions (of type *S* for standard DN) are listed for each teleset the agent is assigned to, or for each hoteling teleset.

  If the agent is using a computer associated with a hoteling teleset, then the extensions of this teleset are shown in this field.

  NOTE: After you choose a different extension, you must log out and log in again in order for the new extension to be in effect.

  Administrators can find more information under "Specifying Telesets" on page 58.

- **Configuration.** This option specifies which communications configuration is in effect for the agent.

  This field lists all configurations with which the agent has been associated. If this option has not previously been chosen, then the default configuration displayed here is the agent's primary configuration, as specified by the administrator. Otherwise, the configuration displayed is the last configuration used.

  Although an agent can be specified for multiple configurations, only one configuration can be in effect at one time.

  NOTE: After you choose a different configuration, you must log out and log in again in order for the new configuration to be in effect. Just before you log out to end a session for any configuration that is not your regular configuration, change this field back to the configuration you regularly use.

  Administrators can find more information under "Specifying Agents" on page 55.

Logging In to or Out of an ACD Queue

This topic describes how to log in to or out of an ACD queue. This procedure is performed in the Communications options of the User Preferences screen.

NOTE: This topic applies only to users who handle voice calls (Siebel CTI users) and receive inbound calls routed from one or more ACD queues. Logging in to ACD queues is not always necessary for voice call implementations, depending on the switch type. Consult your administrator.

In the Agent Queues list in the User Preferences screen, you can log in to or out of individual queues selectively, including those which are not designated as primary.

Alternatively, by using the Log In and Log Out buttons on the communications toolbar, you can log in to or out of all ACD queues designated as primary queues for you, or log in to or out of other communications systems. For more information, see "Logging In to and Out of the Communications System" on page 226, which is part of "Using the Communications Toolbar" on page 223.

Administrators can also configure automatic login and logout. For more information, see "Configuring Communications Login and Logout" on page 172.

To log in to or out of an ACD queue

1. From the application-level menu, choose Tools, then User Preferences.

   The User Preferences screen is displayed.

2. From the link bar, choose Communications.

   The Agent Queues list appears, below the Communications form. All ACD queues with which the agent is associated are listed, and the current login status for each queue is indicated.

3. To log in to a queue, select the record for the queue, then click Login in the Agent Queues list.
4. To log out of a queue, select the record for the queue, then click Logout in the Agent Queues list.

Using the Communications Toolbar

The communications toolbar (which is also called the CTI toolbar) allows you to manage many types of inbound and outbound communications work items. Channel types can include voice, email, fax, wireless message, and others.

Supported channel types depend on the Siebel and third-party products your company is using and on your job function. Your company might support types of communications work items not listed here.

The toolbar buttons can be configured based upon the Siebel modules and third-party communications systems available to you. The communications toolbar is used by various Siebel modules that support communications functions.

The Siebel administrator or configurator can customize the communications toolbar, such as to modify the command a button invokes or add buttons for new commands. For more information, see "About Communications Toolbar Configuration" on page 135.

The agent can use a menu command to refresh the status of the communications toolbar. For more information, see "Using the Menu Command for Refreshing the Communications Toolbar" on page 234.

NOTE: If a record is being created or edited, and has not been committed, then clicking a button on the communications toolbar might not perform the function correctly. You cannot save the record if some required fields have not yet been filled in appropriately. Before performing an action such as to transfer a call (and send a screen transfer), agents must complete and commit all changes or undo the record, as appropriate.

This topic contains the following information:

- "Displaying the Communications Toolbar" on page 223
- "Communications Toolbar Controls" on page 224
- "Logging In to and Out of the Communications System" on page 226
- "Receiving Inbound Work Items" on page 226
- "Initiating Work Items" on page 227
- "Transferring or Conferencing Work Items" on page 228
- "Pausing and Resuming Work Items" on page 230
- "Forwarding Work Items" on page 230
- "Changing the Ready State" on page 231

Displaying the Communications Toolbar

If your Siebel user name has been included as an agent for a communications configuration, then the communications toolbar appears near the top of the browser window containing the Siebel application.

An icon on the application toolbar, Toggle CTI Toolbar, lets you display or hide the communications toolbar. If the communications toolbar is displayed, click the Toggle CTI Toolbar icon to hide the toolbar. If the communications toolbar is hidden, click this icon to display the toolbar.

You can optionally display the communications toolbar as a separate panel that you can move to any location in the browser window. To display the communications toolbar in this way, point to the Toggle CTI Toolbar icon when the communications toolbar is hidden. To move the communications toolbar, drag it to a new location. To hide the communications toolbar again, click Toggle CTI Toolbar or click outside the toolbar panel.

NOTE: The Toggle CTI Toolbar icon and the functionality described in this topic are supported only for deployments using Siebel Open UI.

For more information about using the application toolbar with Siebel Open UI, see *Siebel Fundamentals for Siebel Open UI*.

Related Books

Version 8.1.1.x *Siebel Maintenance Release Guide* on My Oracle Support (Article ID 880452.1)

Version 8.2.2.x *Siebel Maintenance Release Guide* on My Oracle Support (Article ID 1441523.1)

Siebel Fundamentals for Siebel Open UI

Configuring Siebel Open UI

Communications Toolbar Controls

The communications toolbar controls work together to allow you to perform communications tasks within your Siebel application. The toolbar controls have ToolTips, so you can point to any control to learn its name or to determine its current state or availability:

NOTE: All communications features in the Siebel application user interface are subject to the configuration decisions for your company's implementation.

- **Channel Type Indicator.** Indicates the channel type of the active work item, such as email, phone, and so on. If there are no active work items, then the icon is blank.
- **In-Queue Time Indicator.** Indicates, in the ToolTip text, how long the active work item has been assigned to you (in-queue time).
- **Elapsed Time Indicator.** Indicates the working time for the active work item, in the form *HH*:*MM*:*SS*, where *HH* is the hours, *MM* is the minutes, and *SS* is the seconds. This is the time elapsed since a work item became active, such as when an inbound call was answered.
- **Text Input Field.** Allows you to enter data such as the extension of a person to whom you want to transfer a voice call or the extension of a person to whom you want to make a new call. In some hoteling implementations, you might also enter the local computer's host name in this field before logging into the communications system.

  NOTE: Keyboard shortcuts for communications commands do not use as input any data entered into this field.

- **Initiate Work Item.** Allows you to initiate a communications work item. Use the pop-up menu to initiate a work item of a given channel:
  - Make Call (voice or phone channel)
  - Send Email (email channel)
  - Send Fax (fax channel)
  - Send Wireless Message (wireless message channel)
  - Send Page (pager channel)
- **Accept Work Item.** Indicates the channel type of an active inbound communications work item that has arrived. You can click the main button to accept the work item, or use the pop-up menu.
  - Accept Call (voice or phone channel)
  - Accept Email (email channel)
  - Accept Chat (chat channel)
  - A multichannel icon indicates that work items of multiple channels have arrived, such as a voice call and an email message. Use the pop-up menu to accept an individual work item of a specific channel.
- **Release Work Item.** Allows you to release or disconnect the active work item.
- **Blind Transfer.** Allows you to perform a one-step transfer of a work item such as a voice call.
- **Consultative Transfer.** Allows you to initiate and to complete a two-step transfer of a voice call.
- **Conference.** Allows you to initiate and to complete a conference call.
- **Retrieve Call.** Allows you to retrieve the original call when the intended recipient declines a consultative transfer or conference call.
- **Pause Work Item.** Allows you to pause the active work item, such as to put a voice call on hold.
- **Work Items list.** Allows you to select a paused work item on which to resume work.
- **Resume Work Item.** Allows you to resume work on a paused work item selected from the Work Items list.
- **Forward Work Items or Cancel Forward.** Allows you to forward inbound work items, such as forwarding voice calls to a different extension, and to cancel forwarding.
- **Change Ready State.** For each supported inbound channel type, allows you to specify that you are unavailable to accept work items (Not Ready state), or that you are available again (Ready state). Use the pop-up menu to change the state for a specific channel. For example:
  - Ready or Not Ready for Call (voice or phone channel)
  - Ready or Not Ready for Email (email channel)
- **Log In.** Allows you to log in to a communications system such as an ACD queue.
- **Log Out.** Allows you to log out of a communications system such as an ACD queue.

Logging In to and Out of the Communications System

If this capability is supported by your company’s call center implementation, then you can log in to and out of elements of your communications system by using the Log In and Log Out buttons.

Your Siebel application can be configured to log you into ACD call queues and can be configured to support automatic login or automatic logout. For voice calls, these capabilities depend on the CTI middleware your company uses.

If the option Auto Login to Call Center at Startup is checked in your communications user preferences, then starting the Siebel application automatically logs you in. Alternatively, automatic login can be set by the administrator in the communications configuration.

If automatic logout is set by the administrator, then exiting the Siebel application automatically logs you out of applicable communications systems, such as ACD queues.

Use Log In and Log Out commands and the Auto Login to Call Center at Startup setting as instructed by your call center manager.

For more information about the Auto Login to Call Center at Startup user preference, see “Preference Settings for Communications” on page 218.

For information about logging in to or out of ACD queues selectively, see “Logging In to or Out of an ACD Queue” on page 222.

Logging In to the Communications System

This procedure describes how to log in to the communications system.

To log in to the communications system manually

- After starting the Siebel application, click Log In.

Logging Out of the Communications System

This procedure describes how to log in out of the communications system.

To log out of the communications system

- Click Log Out.

Receiving Inbound Work Items

When an inbound work item is routed to you, the Accept Work Item button blinks, and you can accept and begin working on the item.

The Accept Work Item button shows what type of work item is coming in: a phone icon represents a voice call, an envelope icon represents an email message, and so on.

When an inbound work item arrives, the following events might also occur:

- Customer data might be displayed in the customer dashboard.
- A screen pop with relevant data might appear, or you might have to navigate to the appropriate part of the Siebel application.

Accepting an Inbound Work Item

This procedure describes how to accept an inbound work item.

To accept an inbound work item

- Click Accept Work Item when the pause blinks to indicate the arrival of a work item.

You can have multiple work items at the same time, such as voice calls on hold or other paused work items. However, only one work item is active at a time. The channel type of the active work item is indicated on the left side of the communications toolbar.

Releasing an Inbound Work Item

This procedure describes how to release an inbound work item.

To release the current work item

1. Select the work item from the Work Items list.
2. Click Release Work Item, as appropriate for the channel type for the work item.

NOTE: If you are working with an activity record or another type of record that is automatically associated with a work item, then you must save any changes to the record *before* you release the work item. In addition, in order to make changes after releasing the work item, you must first refresh the record. Otherwise, changes you make to the record might be lost.

Initiating Work Items

You can initiate communications work items, such as voice calls or email messages, from the communications toolbar. The Initiate Work Item button allows you to choose the channel type for the work item.

NOTE: Using the Send Email, Send Fax, Send Wireless Message, or Send Page commands in the File menu is equivalent to using the communications toolbar to initiate these activities. For more information about these actions, see *Siebel Email Administration Guide*.

Initiating a Work Item

This procedure describes how to initiate a work item.

To initiate a work item

1. Optionally, specify the recipient for the work item by selecting a contact or employee record, or by entering a phone number into the text input field.

   NOTE: Recipient information entered into the text input field of the communications toolbar applies to initiating voice calls only. It does not apply to initiating email, fax, wireless, or page messages.

2. Do one of the following:

   - If no work item has been initiated within the current session, then click Initiate Work Item to initiate a work item of the channel type that corresponds to the current context.
   - If a work item of a particular channel was previously initiated, then click the Initiate Work Item button to initiate another work item of the same channel type.
   - If you want to choose the channel type explicitly, then click the arrow to the right of Initiate Work Item, then select a supported channel type from the displayed menu.
   - In some cases, you can initiate a voice call to a contact by clicking a hyperlink phone number in a list or by clicking a phone button next to a phone number in a form. This action does not use the communications toolbar.

3. If you did not specify contact information first, then do so now in the dialog box that appears.

Releasing the Current Work Item

This procedure describes how to release the current work item.

To release the current work item

1. Select the work item from the Work Items list.
2. Click Release Work Item, as appropriate for the channel type for the work item.

NOTE: If you are working with an activity record or another type of record that is automatically associated with a work item, then you must save any changes to the record *before* you release the work item. In addition, in order to make changes after releasing the work item, you must first refresh the record. Otherwise, changes you make to the record might be lost.

Transferring or Conferencing Work Items

You can transfer a current work item to another person (such as when you want to escalate an item to a supervisor, or conference another person into a customer call). A voice call can also be transferred to an ACD call queue, depending on your call center. Three transfer and conference operations are supported:

- Blind transfer (one-step transfer)

- Consultative transfer (two-step transfer, voice calls only)
- Conference (two steps, voice calls only)

NOTE: If a screen transfer of the transferring agent’s current record is sent to the receiving agent of a transferred work item, then any changes to this record must be saved before the agent performs the transfer. If a single record only is displayed in a list, then the transferring agent must save changes explicitly before transferring.

Performing a Blind Transfer on the Active Work Item

This procedure describes how to perform a blind transfer on the active work item.

To perform a blind transfer on the active work item

1. Specify the person to whom you want to transfer the active work item by selecting an employee record or by entering contact information, such as phone extension, in the text input field.
2. Click Blind Transfer.

   The work item is released immediately.

Performing a Consultative Transfer on the Active Voice Call

This procedure describes how to perform a consultative transfer on the active voice call.

To perform a consultative (two-step) transfer on the active voice call

1. Specify the person to whom you want to transfer the active voice call by selecting an employee record or by entering the recipient’s extension in the text input field.
2. Click Consultative Transfer.

   The current call is paused, and the transfer recipient is dialed.
3. Do one of the following:
   - When the transfer recipient answers and indicates acceptance of the transfer, click Consultative Transfer again (toggle the button) to complete the transfer and release the call.
   - If the transfer recipient does not accept the transfer, then click Retrieve Call to retrieve the call.

Creating a Conference for the Active Voice Call

This procedure describes how to create a conference for the active voice call.

To create a conference for the active voice call

1. Specify the person with whom you want to conference the active voice call by selecting an employee record or by entering the recipient’s extension in the text input field.

2 Click Conference.

   The current call is paused, and the conference recipient is dialed.

3 Do one of the following:

   - When the conference recipient answers and indicates acceptance of the conference, click Conference again (toggle the button) to complete the conference and include all parties on the active call.
   - If the conference recipient does not accept the conference, then click Retrieve Call to retrieve the call.

4 Repeat Step 1 through Step 3 to add participants, as appropriate.

Pausing and Resuming Work Items

You can pause the active work item, such as putting a voice call on hold, and you can resume a paused work item.

Pausing the Active Work Item

This procedure describes how to pause the active work item.

To pause the active work item

- Click Pause Work Item.

  A voice call is put on hold, or a work item of another channel is paused, until you resume it.

Resuming a Paused Work Item

This procedure describes how to resume a paused work item.

To resume a paused work item

1. If you have an active work item, then pause it, as described earlier.
2. Select a work item from the Work Items list.
3. Click Resume Work Item to activate the selected work item.

Forwarding Work Items

You can have inbound work items of supported channel types forwarded to you, such as having voice calls forwarded to you at a different extension. Only work items subsequently routed to you are forwarded. The toolbar button is a toggle, where the button name changes depending on the state.

To specify forwarding for inbound work items

1. In the text input field, enter the contact information, such as a phone extension, where you want your work items forwarded.
2. Click Forward Work Items.
3. Toggle the button to cancel forwarding.

Changing the Ready State

You can specify that you are unavailable to receive new inbound work items of each supported channel type (such as when you need time to complete after-call work, or you are taking a break). For each channel type, new work items are not assigned to you while the Not Ready state is set for that channel.

You can choose Change Ready State separately for each channel type, or for work items of all supported channel types. The toolbar button is a toggle.

If all channels are in Not Ready state, then the button is toggled down (appearing to be depressed). If any, or all, channels are in Ready state, then the button is toggled up.

Indicating That You Are Not Ready to Receive Inbound Work Items

This procedure describes how to indicate that you are not ready to receive inbound work items.

To indicate that you are not ready to receive inbound work items

- Do one of the following:
  - Click Change Ready State to indicate that you are not ready to receive any inbound work items of all channel types for which you are eligible.
  - Click the arrow to the right of Change Ready State, then choose one or more options from the displayed menu to indicate that you are not ready to receive work items for the applicable channel types.

NOTE: If you click the Change Ready State button to set the Not Ready State for all channel types, but the button does not display as toggled-down (depressed), then this might be due to a configuration error or some other issue for one or more channels. Report this issue to your supervisor. You can still choose the Not Ready State for individual channels, as described in the second bullet item.

Indicating That You Are Ready to Receive Inbound Work Items

This procedure describes how to indicate that you are ready to receive inbound work items.

To indicate that you are ready to receive inbound work items

- Do one of the following:
  - Click Change Ready State to indicate that you are ready to receive any inbound work items of all channel types for which you are eligible.
  - Click the arrow to the right of Change Ready State, then choose one or more options from the displayed menu to indicate that you are again ready to receive work items of the applicable channel types.

Using Communications Menu Commands

This topic describes using the Communications submenu and communications commands in the applet-level menus. It contains the following information:

- "Using the Menu Commands for Displaying Error Messages" on page 233
- "Using the Menu Command for Refreshing the Communications Toolbar" on page 234

The Communications submenu, which is a submenu of the Tools application-level menu in a communications-enabled Siebel application, contains several communications commands for agents to use. The applet-level menus also contain communications commands. Some of the menu commands might perform functions equivalent to buttons in the communications toolbar.

The communications-related menu options support context sensitivity. Agents can initiate several communications operations that extract data from Siebel database records, such as phone numbers for contacts or service requests, in order to initiate communications or take actions affecting the current communications work item.

NOTE: If a record is being created or edited and has not been committed, then choosing a communications menu command might not perform the function correctly. You cannot save the record if some required fields have not yet been filled in appropriately. Before performing an action such as to transfer a call (and send a screen transfer), agents must complete and commit all changes or undo the record, as appropriate.

To choose communications commands

- Do one of the following:
  - From the application-level menus, choose Tools, then Communications, then one of the displayed submenu commands.
  - From the applet-level menu, choose one of the displayed communications commands.

Example commands might include Answer Call, Associate, Blind Transfer, View Work Item, and so on.

The available commands in these menus vary according to the communications configuration in effect for the agent. Administrators can customize the content and functioning of the Communications submenu and the applet-level menus by working with the commands in the All Commands view in the Administration - Communications screen. For more information, see "Configuring Communications Menu Commands" on page 145.

NOTE: If you are working with an activity record or another type of record that is automatically associated with a work item, then you must save any changes to the record *before* you release the work item. In addition, in order to make changes after releasing the work item, you must first refresh the record. Otherwise, changes you make to the record might be lost.

Using the Menu Commands for Displaying Error Messages

Agents can choose the following menu commands in order to redisplay messages, such as error messages, that had previously appeared in the area to the right of the application-level menus. These commands might be helpful for troubleshooting purposes. These commands are available through the communications configurations provided with Siebel Business Applications.

Displaying the Previous Message

This procedure describes how to display the previous message.

To display the previous message

- From the application-level menu, choose Tools, then Communications, then Toolbar, and then Previous Message.

  The previous message appears, from the stack of messages that had appeared since the time the agent logged on.

Displaying the Next Message

This procedure describes how to display the next message.

To display the next message

- From the application-level menu, choose Tools, then Communications, then Toolbar, and then Next Message.

  The next message appears, assuming the agent previously invoked the Previous Message command.

Using the Menu Command for Refreshing the Communications Toolbar

Agents can choose the following menu command in order to refresh the status of buttons on the communications toolbar. (The message area to the right of the application-level menus is not refreshed.)

To refresh the communications toolbar

- From the application-level menu, choose Tools, then Communications, then Toolbar, and then Refresh.

  The communications toolbar is refreshed to the current status.

10 Using the Virtual CTI Driver

This chapter describes how to use the Virtual CTI driver for Siebel Communications Server to support Oracle Contact On Demand for Siebel Business Applications. It includes the following topics:

About the Virtual CTI Driver

The Virtual CTI driver is a communications driver you can use to integrate your Siebel Business Applications with Oracle Contact On Demand. With this type of deployment, agents can access communications functionality using the communications toolbar. See also Chapter 2, "Overview of Siebel CTI and Related Products," and related content in this guide. Oracle Contact On Demand documentation is located on Oracle Technology Network.

Process of Integrating with the Virtual CTI Driver

To implement an integration using the Virtual CTI driver, you must carry out the following tasks:

1. Configure the Virtual CTI driver.

   For more information, see "Virtual CTI Driver Parameters" on page 236.

   See also Chapter 3, "Configuring Communications Drivers and Profiles."

2. Create a communications configuration for Oracle Contact On Demand and the Virtual CTI driver.

   For more information about creating communications configurations, Chapter 4, "Configuring Siebel CTI," including "Creating or Modifying a Communications Configuration" on page 44.

   For information about commands and events for this configuration, see "Virtual CTI Driver Commands" on page 244 and "Virtual CTI Driver Events" on page 253. See also Chapter 5, "Configuring Events and Commands."

3. Specify agents and telesets for your communications configuration. Perform the following administration steps:

   a. Enter the Oracle Contact On Demand agent's Agent Login and Password for each Siebel agent.

   b. Associate one dummy teleset with each agent.

For more information, see Chapter 4, “Configuring Siebel CTI,” including “Relationship of Agents and Telesets for Siebel CTI” on page 55.

NOTE: The Virtual CTI driver can only be used on the Microsoft Windows platform.

Virtual CTI Driver Parameters

Table 50 describes the supported driver parameters specific to the Virtual CTI driver which enables you to connect your Siebel Communications Server to other applications and their functionality.

You view and modify these driver parameters in the Communications Drivers and Profiles view, which is one of the views in the Administration - Communications screen.

For descriptions of macros referenced in the table, such as @Username, see “Macros for Parameter Values” on page 152.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req’d	Default Value	Description
Driver: AdminPassword	Yes	Not applicable	Enter the password of the administrator for the application.
Driver: AdminUserName	Yes	Not applicable	Enter the user name of the administrator for the application.
Driver: AutoProxyRefreshInterval	No	-1	Specify the interval in seconds before the Virtual CTI driver requests a refreshed list of proxy servers. You set a value for this parameter if Driver:ProxySupport is set to Auto or Script. The default value is -1. When set to this value, the Virtual CTI driver retrieves the list of proxy servers once for each driver session. The minimum value that you can specify as an interval is 60 seconds. If you specify a value less than 60 seconds, then the Virtual CTI driver resets it to 60 seconds.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: BackupURL	No	Not applicable	The Oracle Contact On Demand server can fail over to another server using a backup connection. Use this parameter to specify a backup Oracle Contact On Demand server. If the primary server (configured using Driver:URL) goes down, then the Virtual CTI driver switches connections to the backup server. The logic of this feature is: ■ For a new agent login, try the primary server first. If it fails, then switch to the backup server. ■ For an agent already logged in, if the working server (primary or backup) fails, then switch to the other one. ■ If a connection cannot be made to either the primary or the backup server, then the driver does not try to reconnect and goes into a disabled state.
Driver: CompanyName	Yes	Not applicable	Enter the name of the company that has licensed the application.
Driver: CountryCodeDelimiter	No	. (period)	Specify the delimiter to use between a country code and local area code if an agent enters a phone number in the communications toolbar. The default delimiter is a period (.). See also the description of the CountryCodeDelimiter configuration parameter, in "Parameters for Communications Configurations" on page 47.
Driver: DefaultCountryCode	No	1	Specify the default country code if it is not specified by the agent.
Driver: EnableSOAPTrace	No	False	Set this parameter to True to enable logging for communication using the SOAP protocol. If set to True, then the driver writes entries to the log file specified by Driver:LogSOAPFile.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: EventRetry	No	50	The number of attempts that the driver makes when getting event failed messages after each SOAP request. The total number of attempts for processing such SOAP requests is: SOAPRetry*EventRetry. If the request still fails after this number of attempts, then the driver stops and all agents inside are logged out automatically. The valid range of values is 1 to 1000.
Driver: HotellingSupport	No	False	Specify whether to enable or disable the hoteling feature. Set this parameter to True to enable hoteling. In order to use hoteling, also define command data for a login command (for example, LoginToPBX) using at least one of the following macros: ■ @ClientHostName ■ @ClientIP For more information about using macros, see "Macros for Parameter Values" on page 152.
Driver: LogFile	No	tw.log	Specifies the file name of the log file for the driver. See also the description for the Service:LogFile parameter.
Driver: LogSOAPFile	No	twsoap.log	Specifies the file name of the SOAP log file for the driver. The Driver:EnableSOAPTrace parameter must be set to True before the driver writes to this log file.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: MaxAgentNumber	No	-1	Specifies the maximum number of agents that the driver instance can handle. When any additional agents log in (using the same communications configuration), a new instance of the driver is created. For any value less than 1 (including the default value of -1), one driver instance handles all agents using the same communications configuration.
Driver: MaxMessageNumber	No	50	Specifies the maximum number of messages that the driver retrieves with each call.
Driver: MessageInterval	No	1000	Specify the interval in milliseconds between each request by the driver to retrieve messages.
Driver: OnHookAtLogout	No	True	If True, then set the teleset used by the agent to available once the agent logs out of the application.
Driver: ProxyLocation	No	Not applicable	You must specify a value for this parameter if you set the value of the Driver:ProxySupport parameter to Script or Static. Otherwise, no parameter value is required. Specify a valid URL in a Unicode string format that points to the location of your network's proxy auto-configuration (PAC) file. For example, if you use a script, then the URL might be as follows: http://wpad.example.com/wpad.dat or http://www.example.com/wpad.js If you use a static location, then the URL would be of the form: http://www.example.com:*port* Contact your network administrator to obtain the location of the PAC file. **NOTE:** Oracle supports only ECMAScript-based PAC files.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: ProxySupport	No	Disabled	Set a value for this parameter to specify the driver's support for proxy servers. Values can be one of these: ■ **Disabled (default value).** The driver does not support proxy servers. ■ **Auto.** The driver retrieves a list of proxy servers by executing a PAC file. It executes the file by issuing a dynamic host configuration protocol (DHCP) request or by searching for a DNS address in the network. ■ **Script.** The driver retrieves a list of proxy servers by executing a PAC file. You specify the location of the file by entering a value for the Driver:ProxyLocation parameter. For the file location, contact your network administrator. ■ **Static.** The driver uses the proxy server that you specify. When you set Driver:ProxySupport to Auto or Script, the driver uses the first proxy server in the retrieved list. If this proxy server fails to respond, then the driver does not try to use the next working proxy server in the list. Instead, the driver retrieves the list again after the interval specified by the Driver:AutoProxyRefreshInterval parameter expires. This process repeats until a valid proxy server is retrieved or the maximum SOAP requests specified by the parameter Driver:SOAPRetry is reached.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: RefreshCCALibraries Interval	No	600 (seconds)	Specify the interval in seconds before the Virtual CTI driver refreshes the library data in the driver cache. To use this parameter, set a value to 60 or more. (If a value less than 60 is specified, then the effective value is 60.)
Driver: SOAPRetry	No	3	Enter the maximum number of times to attempt a SOAP request before failing with an error code. The valid range of values is 1 to 20.
Driver: UnknownCCAValue	No	Not applicable	Specifies a default value to display when a nonexistent field is specified for the parameter Driver:VoiceDescription. For example, assume: Driver:VoiceDescription = [ani]/[DNIS]/[workgroupName] Driver:UnknownCCAValue = ?? Building upon the example for Driver:VoiceDescription, when an agent gets the incoming ACD call, the message displayed on the communications toolbar would be *Voice call from ??/4321/VoiceCallGroup*. The string *??* is displayed because *ani* is not a valid field name that would be received from Oracle Contact On Demand. (The correct field name would be *ANI*, not *ani*.) This parameter applies only for a nonexistent field. It does not substitute the empty value of an existent field. See also the parameter description for Driver:VoiceDescription.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Driver: URL	Yes	Not applicable	Enter the URL of the application with which you are integrating. Specify the URL of Oracle Contact On Demand's Web service here to enable the driver to connect to it.
Driver: VoiceDescription	No	Not applicable	A work item description template for an incoming ACD call. Brackets enclose field identifiers, so the parameter value might resemble this example: [ANI]/[DNIS]/[workgroupName] where ANI, DNIS, and workgroupName are field names. Each field name is substituted with a value obtained at run time from Oracle Contact On Demand. **NOTE:** Field names are case-sensitive and must exactly match the data received from Oracle Contact On Demand. For example, assume an incoming call where ANI equals 1234, DNIS equals 4321, and workgroupName equals VoiceCallGroup, and where Driver:VoiceDescription equals [ANI]/[DNIS]/[workgroupName]. The communications toolbar displays *Voice call from 1234/4321/ VoiceCallGroup*. See also the parameter description for Driver:UnknownCCAValue.
LogDebug	No	True	When set to True, the driver writes to the log file specified by the Driver:LogFile parameter, the Service:ServiceLogFile parameter, or both. If you specify this parameter prefixed with either Service: or Driver:, then it applies to the applicable log file only.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
MaxLogKB	No	90240	Specifies the maximum size of the log file, in kilobytes (KB). This parameter is generic and applies to log files specified using Driver:DriverLogFile and using Service:ServiceLogFile. If you specify this parameter prefixed with either Service: or Driver:, then it applies to the applicable log file only. When the applicable log file is full, it is emptied completely, then logging begins again.
ReleaseLogHandle	No	True	Indicates that the log file handle for each agent is released periodically, after logs have been generated. The default setting of True provides better performance. When ReleaseLogHandle is False, however, some log files might not be generated if the number of concurrent users exceeds the file-handle capacity provided by the operating system. This parameter is generic and applies to log files specified using Driver:LogFile and using Service:LogFile. If you specify this parameter prefixed with either Service: or Driver:, then it applies to the applicable log file only.

Table 50. Virtual CTI Driver Parameters

Parameter Name	Req'd	Default Value	Description
Service: LogFile	No	tw_{@Username}.log	If a filename is specified using this parameter, then Siebel Communications Server writes detailed information about activity for the active call, for each agent's session, into a log file. Data for Siebel Communications Server. Driver: activity is logged, including device events and event data fields. Log files are created by default in the log subdirectory of the Siebel Server installation directory, depending on the client type and on your communications deployment. See also the description for the Driver:LogFile parameter.

Virtual CTI Driver Commands

Table 51 describes the commands that the Virtual CTI driver supports.

NOTE: For the Virtual CTI driver commands, the TrackingID parameter is used to track the work item and is therefore optional.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
AcceptChatWork	Not applicable	Accepts a new inbound chat and puts the active chat session on hold if an active chat session exists.
AcceptEmailWork	Not applicable	Accepts a new inbound email.
AnswerCall	Not applicable	Accepts a new inbound call and puts the active call on hold if an active call exists.
ChangeNotReadyState	Reason	Toggles the agent ready status, where Reason references the reason code number. The reason code number is the agent status ID specified at Oracle Contact On Demand configuration.
ConferenceComplete	Not applicable	Completes a conference previously initiated, so that all parties are joined in the call.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
ConferenceInit	PhoneNumber	Initiates a conference call between the active call and the extension or external number specified by PhoneNumber.
ConferenceInitExtension	PhoneNumber	Initiates a conference call between the active call and the extension number specified by PhoneNumber.
ConferenceInitExternal	PhoneNumber	Initiates a conference call between the active call and the external number specified by PhoneNumber.
DeclineCall	Not applicable	Declines a new inbound call. The agent cannot decline an internal call from another agent.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description

DoTransfer	WorkgroupID	Transfers the selected work item to the workgroup specified by WorkgroupID. Workgroup IDs are defined during Oracle Contact On Demand configuration.
		
		Workgroup IDs can be obtained in one of two ways. One method is explained in the description for the command GetTransferList.
		
		Alternatively, you can define Lists of Values (LOVs) for workgroups in the Siebel applications. First, create a List of Values type, such as *CCOD CTI Workgroup*. Then create a corresponding LOV of this type, where the Language-Independent Code value matches the workgroup ID.
		
		For example, you might define command and command data definitions like the following. Make sure the Name field for the LOV corresponds to the name of the workgroup in Oracle Contact On Demand.
		
		[Command:BlindTransferVoice]
		Description = "Blind transfer voice to workgroup"
		DeviceCommand = "DoTransfer"
		FilterSpec = "[@SelectedWorkItem:mediaType] IS NULL"
		Hidden = "True"
		Comments "For Voice workgroup transfer"
		CmdData = "BlindTransferVoice"
		
		[CmdData:BlindTransferVoice]
		AttachContext = "True"
		Param.Data = "[Name]"
		Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"
		SelectApplet = "Transfer Multiple LOV Popup Applet"
		SelectBusComp = "List Of Values"
		SelectBusObj = "List Of Values"
		SelectParam = "True"
		SelectQuerySpec = "[Type] = 'CCOD CTI Workgroup' AND [Active] = 'Y'"
		SelectTitle = "Begin Blind Transfer Voice to:"

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
EndWrapUp	InteractionID	Ends wrap-up mode for a work item, where InteractionID references the third-party application's unique ID for the work item. The agent must be configured for wrap-up mode through Oracle Contact On Demand for this command to be enabled. The default wrap-up time is also configured through Oracle Contact On Demand. Wrap-up mode begins when an agent releases a work item (disconnecting the customer). No new work items are routed to the agent until wrap-up mode is ended.
EnterConference	Not applicable	Enters a conference.
GetMediaFile	InteractionID	Gets the media file referenced by the InteractionID. The output is a temporary URL that links to the media file.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description

GetTransferList	Not applicable	Gets a list corresponding to workgroups defined in Oracle Contact On Demand. The output is a property set containing the workgroup list, where the key is the workgroup name and the value is the workgroup ID.
		
		The workgroup name and workgroup ID correspond to the DisplayValue and InternalValue fields in the virtual business component TaW PopList (class name CSSBCVTaWPopList). This virtual business component invokes the GetTransferList command internally.
		
		The applet TaW Transfer Applet maps to the virtual business component TaW PopList. You can use a command data definition (for a command invoking DoTransfer, for example) like the following to obtain the workgroup list:
		
		[CmdData:VoiceTransferPop]
		SelectApplet = "TaW Transfer Applet"
		SelectBusComp = "TaW PopList"
		SelectBusObj = "CommSrv CM Teleset Administration"
		SelectParam = "True"
		Param.Data = "[InternalValue]"
		Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"

HoldCall	TrackingID	Puts the active work item or wrap-up work on hold.
LeaveConference	Not applicable	Exits the conference.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
LogIn	■ AgentID ■ AgentPin ■ PhoneNumber	Allows the agent to log in and uses the number specified by PhoneNumber. AgentID specifies the agent's user ID in the application that the driver connects to while AgentPin specifies the corresponding password. Valid formats for PhoneNumber are as follows: ■ Phone number without country code The driver assigns the default country code to the number. For example, if the default country code is 86 (for China) and PhoneNumber is: *xxxxxxxxx* then the driver makes an outbound call to the following number: 86.*xxxxxxxxx* ■ *CountrycodeDelimiterPhonenumber* The following example parameter specifies a number in India: 91.*xxxxxxxxxx* where: ■ 91 is the country code (for India) ■ . is the country code delimiter ■ *xxxxxxxxxx* is the phone number to call ■ SIP:*IP_address* The following is a valid example: SIP:10.182.106.120
LogOut	Not applicable	Logs out the agent from the current session.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
MakeCall	PhoneNumber	Makes an outbound call using the value of the PhoneNumber parameter as input. Valid formats for PhoneNumber are as follows: ■ Phone number without country code The driver assigns the default country code to the number. For example, if the default country code is 86 (for China) and PhoneNumber is: *xxxxxxxxx* then the driver makes an outbound call to the following number: 86.*xxxxxxxxx* ■ *CountrycodeDelimiterPhonenumber* The following example parameter specifies a number in India: 91.*xxxxxxxxxx* where: ■ 91 is the country code (for India) ■ . is the country code delimiter ■ *xxxxxxxxxx* is the phone number to call ■ X:*ExtensionNumber* The following example parameter specifies an extension number, 1234, to call: X:1234
MakeExtensionCall	PhoneNumber	Makes an internal call to the number specified by PhoneNumber. The following example shows a valid value for PhoneNumber: 1234
MakeExternalCall	CountryCode: PhoneNumber	Makes an outbound call to the specified phone number. You can optionally include a CountryCode. If you do not include CountryCode, then the default CountryCode is used.
OffHook	Not applicable	Sets the phone off hook.

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
OnHook	Not applicable	Sets the phone on hook.
PushURL	■ InteractionID ■ Data	Sends a URL to the customer's chat window, where Data specifies a valid URL and InteractionID is a third party's unique ID for the work item.
ReleaseCall	TrackingID	Releases the current active call.
RetrieveCall	Not applicable	Retrieves a collaborative or supervised transfer call. Retrieves the original call after the current call has been released using ReleaseCall.
SendChatMessage	■ InteractionID ■ Data	Sends a chat message to the customer's chat window, where Data specifies the chat message string and InteractionID is a third-party product's unique ID for the work item.
SendToConference	InteractionID	Sends the active work item specified by InteractionID to the conference.
SetAgentToNotReady	Reason	Sets the agent status to not ready, where Reason references the reason code number. The reason code number is the agent status ID specified in the Oracle Contact On Demand configuration. If it is not specified, then the default value is 2.
SetAgentToReady	Reason	Sets the agent status to ready, where Reason references the reason for moving to the ready state. The reason code number is the agent status ID specified at Oracle Contact On Demand configuration. If it is not specified, then the default value is 1.
StartRecording	TrackingID	Starts recording the active work item.
StopRecording	TrackingID	Stops recording the active work item.
TransferComplete	Not applicable	Completes a supervised transfer call.
TransferInit	PhoneNumber	Transfers the selected work item to the number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
TransferInitExtension	PhoneNumber	Transfers the selected work item to the extension phone number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"
TransferInitExternal	PhoneNumber	Transfers the selected work item to the external phone number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"
TransferMute	PhoneNumber	Initiates a mute (blind) transfer of the selected work item to the number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"
TransferMuteExtension	PhoneNumber	Initiates a mute (blind) transfer of the selected work item to the extension phone number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"
TransferMuteExternal	PhoneNumber	Initiates a mute (blind) transfer of the selected work item to the external phone number specified by PhoneNumber. The command definition must include the following parameter: Param.TrackingID = "{@SelectedWorkItem:DriverWorkTrackID}"

Table 51. Virtual CTI Driver Commands

Event Name	Parameters	Description
UnHoldCall	TrackingID	For a selected work item or wrap-up work currently on hold, resumes the work item as the active work item or returns to wrap-up work.

Virtual CTI Driver Events

Table 52 describes the events that the Virtual CTI driver supports.

This topic also includes "Details for the InteractionMessage Event" on page 255.

Table 52. Virtual CTI Driver Events

Event Name	Description
AgentMessage	Includes basic information about the agent, such as agent status, login, and logout. Values for the parameter MsgType represent different types of agent messages, as follows: **TYPE_SESSION_CHANGE.** Agent session change **TYPE_LOGIN_SUCCESS.** Agent has successfully logged in **TYPE_LOGIN_AGAIN.** Agent should log in again **TYPE_LOGOUT_BY_SUPERVISOR.** Agent is logged out by the supervisor **TYPE_LOGOUT_BY_SECOND_LOGIN.** Agent is logged out by a second log in **TYPE_STATUS.** A change in the agent's status **TYPE_CONFERENCE.** A change in the agent's conference status **TYPE_PREDICTIVE.** A change in the agent's predictive status **TYPE_LBROADCAST_MESSAGE.** Agent received a broadcast message **TYPE_AGENTS_STATUS.** Agent received status information on all other agents
CustomMessage	A custom message used for sending and receiving driver specific messages, for example, sending bookmark data for a screen transfer. This event type is not used for event handlers.
DataMessage	This event type is not used for event handlers.

Table 52. Virtual CTI Driver Events

Event Name	Description
ErrorMessage	An error-related message from Oracle Contact On Demand, such as an error code and an error message to be displayed. This event type is not used for event handlers.
InteractionMessage	Includes basic information about the interaction, such as the new, active, or end interaction type and chat messages. For more information about how the InteractionMessage event is used, see "Details for the InteractionMessage Event" on page 255.
PhoneMessage	A phone terminal handling message, such as on hook, off hook, or mute. This event type is not generally used for event handlers, but one example is provided. The value LINE_TYPE for the parameter MsgType represents a change in the phone status. Values for the parameter LineState represent different types of phone messages, as follows: **STATE_UNKNOWN.** Unknown phone status **STATE_MUTED.** Phone is at mute status **STATE_NOT_MUTED.** Phone is not a mute status **STATE_OFF_HOOK.** Phone is off hook **STATE_ON_HOOK.** Phone is on hook **STATE_RINGING.** Phone is ringing **STATE_TALKING.** The phone is connected and the agent is talking A partial example using STATE_TALKING is as follows: [EventHandler:PhoneOnTalk] Filter.MsgType = "TYPE_LINE" Filter.LineState = "STATE_TALKING" DeviceEvent = "PhoneMessage" Response = "PhoneOnTalk" [EventResponse:PhoneOnTalk] ...
StatisticMessage	A statistical message, such as agent statistics. This event type is not used for event handlers.

Details for the InteractionMessage Event

This topic provides additional details about the InteractionMessage event.

MsgType Parameter for InteractionMessage

Values for the parameter MsgType represent different types of interaction messages (for the InteractionMessage event), as described in Table 53.

Table 53. MsgType Parameter for InteractionMessage Event

Value	Description
TYPE_NEW	A new interaction message was received, but has not yet been accepted.
TYPE_END	The interaction was ended.
TYPE_STATE	A change in the interaction state.
TYPE_CHAT_MESSAGE	A new chat message was received for an interaction.
TYPE_SMART_RESPONSE	A new smart chat response was received for an interaction.
TYPE_WRAPUP	A wrap-up period started for an interaction.
TYPE_CHAT_TRANSFER	A chat transfer message was received for an interaction.
TYPE_PUSH_URL	A push URL message was received for an interaction of the chat type.
TYPE_INTERACTION_STATE_SYNCHRONIZATION	A message was received regarding synchronization of interaction states.

Channel Parameter for InteractionMessage

Values for the parameter Channel represent different channels for interaction messages, as described in Table 54.

Table 54. Channel Parameter for InteractionMessage Event

Value	Description
Phone	Workgroup call
Chat	Workgroup chat
Email	Workgroup email
Fax	Workgroup fax
Callback	Workgroup callback

Table 54. Channel Parameter for InteractionMessage Event

Value	Description
WebCallback	Workgroup Web callback
VoiceMail	Workgroup voicemail
PredictiveCall	Predictive call
PreviewCall	Preview call
InboundExtension	Inbound extension
OutboundExtension	Outbound extension
InboundCall	Inbound call
OutboundCall	Outbound call
DirectChat	Outbound chat
OutboundEmail	Outbound email
Unknown	Unknown interaction type

InboundCall Parameter for InteractionMessage

Values for the parameter InboundCall represent whether the interaction message is inbound or outbound, as described in Table 55.

Table 55. InboundCall Parameter for InteractionMessage Event

Value	Description
0	An outbound interaction
1	An inbound interaction

SiebelCall Parameter for InteractionMessage

Values for the parameter SiebelCall represent whether or not the interaction message is between agents using Siebel CTI and the Virtual CTI driver, as described in Table 56.

Table 56. SiebelCall Parameter for InteractionMessage Event

Value	Description
0	A regular call from or to an external customer
1	An interaction between agents using Siebel CTI and the Virtual CTI driver

SecondaryOperation Parameter for InteractionMessage

Values for the parameter SecondaryOperation represent whether or not the interaction message is in the middle of a transfer, as described in Table 57.

Table 57. SecondaryOperation Parameter for InteractionMessage Event

Value	Description
0	A regular call (not from another agent using Siebel CTI and the Virtual CTI driver) that is in the middle of a transfer
1	All other types of calls

CallConnected Parameter for InteractionMessage

The value 1 for the parameter CallConnected indicates that the new call (interaction message) has been accepted by the agent.

State Parameter for InteractionMessage

Values for the parameter State represent the state of the interaction message, as described in Table 58.

Table 58. State Parameter for InteractionMessage Event

Value	Description
STATE_UNKNOWN	An unknown interaction state
STATE_NEW	A new interaction
STATE_HOLD	The interaction is on hold
STATE_VOICEMAIL	The interaction is in voicemail
STATE_ACTIVE	The interaction is active
STATE_CONFERENCE	The interaction is in conference
STATE_DISCONNECT	The interaction is disconnected or has ended
STATE_RECORDING_STARTED	Recording was started on the interaction
STATE_RECORDING_STOPPED	Recording was stopped on the interaction

Example Event Handlers for InteractionMessage

Using the various parameters described earlier for the InteractionMessage event, many different kinds of events can be handled. See the following examples for some common event handlers.

NOTE: Many other kinds of event data can be used for an event handler, such as priority, WorkgroupID, ANI, DNIS, and so on. All such data comes from Oracle Contact On Demand. For more information about events that might be received, check the driver log file and see documentation for Oracle Contact On Demand on Oracle Technology Network. See also Table 52 on page 253.

New Inbound Phone Call from an Oracle Contact On Demand Workgroup Arrives

```
[EventHandler:NewInboundCallFromCOD]
Filter.MsgType = "TYPE_NEW"
Filter.Channel = "Phone"
Filter.InboundCall = "1"
Filter.SecondaryOperation = "1"
Filter.SiebelCall = "0"
DeviceEvent = "InteractionMessage"
Response = "NewInboundCallFromCOD"

[EventResponse:NewInboundCallFromCOD]
...
```

New Internal Phone Call from Another Agent Arrives

```
[EventHandler:NewInsideCallFromCOD]
Filter.MsgType = "TYPE_NEW"
Filter.Channel = "Phone"
Filter.InboundCall = "1"
Filter.SecondaryOperation = "1"
Filter.SiebelCall = "1"
DeviceEvent = "InteractionMessage"
Response = "NewInsideCallFromCOD"

[EventResponse:NewInsideCallFromCOD]
...
```

New Inbound Phone Call Is Connected

```
[EventHandler:InboundCallAccepted]
Filter.CallConnected = "1"
Filter.Channel = "Phone"
Filter.InboundCall = "1"
Filter.SecondaryOperation = "1"
DeviceEvent = "InteractionMessage"
Response = "InboundCallAccepted"

[EventResponse:InboundCallAccepted]
...
```

Connected Phone Call Has Been Put on Hold

```
[EventHandler:InteractionOnHold]
Filter.MsgType = "TYPE_STATE"
Filter.State = "STATE_HOLD"
Filter.Channel = "Phone"
DeviceEvent = "InteractionMessage"
Response = "InteractionOnHold"

[EventResponse:InteractionOnHold]
...
```

Connected Phone Call Has Returned to Active Status

```
[EventHandler:InteractionResumed]
Filter.MsgType = "TYPE_STATE"
Filter.State = "STATE_ACTIVE"
FilterSpec = "[CallConnected] IS NULL"
Filter.Channel = "Phone"
DeviceEvent = "InteractionMessage"
Response = "InteractionResumed"

[EventResponse:InteractionResumed]
...
```

Phone Call Has Been Released

```
[EventHandler:InteractionEnd]
Filter.MsgType = "TYPE_END"
Filter.Channel = "Phone"
DeviceEvent = "InteractionMessage"
Response = "InteractionEnd"

[EventResponse:InteractionEnd]
...
```

A Developing a Communications Driver

This appendix provides developer guidelines for writing communications drivers using the Siebel Adaptive Communications application programming interface. It includes the following topics:

Required Skills for Adaptive Communications Developer

To develop an interface to Adaptive Communications, take the time to learn about the following:

- Siebel Business Applications, from an end-user and development standpoint
- Communications concepts and event-handling model
- Technologies and development tools for the languages C++ or C

NOTE: Custom communications drivers are to be supported and maintained by their originators. Oracle is not liable in any way for custom communications drivers.

Custom Driver Upgrade Issues

This topic describes issues that might apply if you are supporting or updating custom communications drivers originally written for a Siebel CRM release 7.x that is earlier than version 7.7; that is, for drivers written for Siebel CRM version 7.0.*x* or version 7.5.*x*.

Communications drivers for prior Siebel CRM 7.*x* releases do not work *as is* with Siebel CRM version 7.7.*x* or later, unless you follow the guidelines in this topic. Siebel CRM version 7.7.*x* or later requires version 2 of the Siebel Adaptive Communications API (as did Siebel CRM version 7.5.*x*). For more information about custom drivers, see "Communications Drivers and Third-Party Systems" on page 32.

Requirements that *might* apply to your custom communications driver are as follows:

- On all supported operating systems, for any communications driver written for Siebel CRM version 7.0.*x*, you *must* compile the driver using the current version of the API. A Siebel CRM version 7.0.*x* communications driver cannot work with Siebel CRM version 7.7.*x* or later, due to version checking. For drivers from Siebel CRM version 7.5.*x*, this requirement is already met.

- For drivers on all supported operating systems, the driver handle method FreeSCStrParamList *must* be implemented. For any driver for Siebel CRM version 7.0.*x* or 7.5.*x*, if you previously did not implement this method to function as described, then you must implement it and recompile your driver. If you previously implemented this method, then this requirement might not apply. For more information about this method, see "Methods of ISC_DRIVER_HANDLE" on page 276.
- For Oracle Solaris, you *must* use UTF16, in order to support Unicode, rather than UCS4 (as was used in Siebel CRM version 7.0.*x*). For any driver on this platform for Siebel CRM version 7.0.x, you must modify your driver code and recompile your driver. For drivers from Siebel CRM version 7.5.*x*, this requirement is already met.

Adaptive Communications Design

Siebel Adaptive Communications is a programmable software layer between Siebel applications and the external communications system that a communications driver is written to support, such as a CTI middleware package or email server.

The Adaptive Communications layer has two main parts:

- The client handle, which is how the communications driver can access functionality in the Siebel application
- The communications driver (encompassing the driver handle and the service handle), through which communications pass between the Siebel application and the external communications system

This topic contains the following information:

- "Communications Drivers" on page 262
- "Adaptive Communications Architecture" on page 263
- "Adaptive Communications Event and Command Model" on page 264
- "Initialization of Communications Drivers" on page 264
- "Driver Parameters and Initialization" on page 266
- "Driver Event Attributes" on page 266

Communications Drivers

A variety of communications drivers are available:

- Oracle provides communications drivers for use with Oracle Contact On Demand (Virtual CTI driver); standard email servers that support SMTP, IMAP, and POP3; and other communications systems. For more information about configuring communications drivers, see Chapter 3, "Configuring Communications Drivers and Profiles."

  For driver parameter, command, and event details for each driver provided with Siebel Business Applications, see the applicable content on each supported driver. For example, for the Virtual CTI driver, see Chapter 10, "Using the Virtual CTI Driver." See also *Siebel Email Administration Guide*. For information about third-party product support, see My Oracle Support Certifications.

- Siebel CRM customers and vendors of CTI middleware, email servers, and so on, can use Adaptive Communications to write custom drivers to support other communications systems. To find out about communications drivers created by other vendors, refer to the Partners page on the Oracle Web site, http://www.oracle.com.

Drivers can be created that run on Microsoft Windows or on UNIX. For example, drivers on Microsoft Windows might be Dynamic Link Libraries (DLL files), or drivers on UNIX operating systems might be shared object (.so) files. Alternatively, drivers can be created as other types of files than these, such as executable files.

Adaptive Communications Architecture

The communications flow among the elements of Adaptive Communications is shown in Figure 3. As shown, the implementations of the driver handle and the service handle together make up the communications driver, along with any other program data and logic built into the driver.

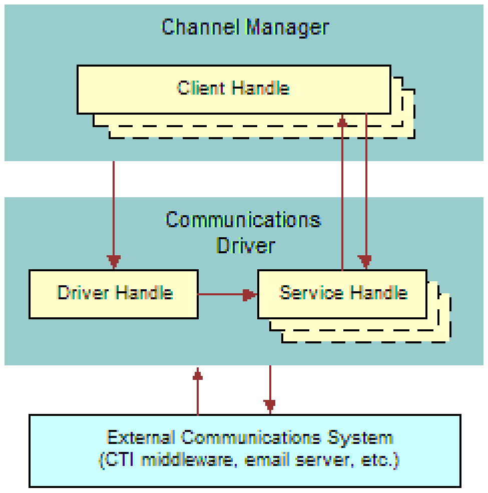

Figure 3. Adaptive Communications Architecture

The channel manager initializes the implementation object for the driver handle, which in turn initializes the implementation object for the service handle.

Multiple instances of the client handle and service handle implementation objects might be initialized, such as for each interactive agent session, depending on the driver implementation and on the client deployment. For an overview of the entire communications system, see “About Siebel Communications Server Architecture” on page 18.

Adaptive Communications Event and Command Model

Siebel Adaptive Communications, when it is used to write interactive drivers, such as those used with CTI middleware, uses the concept of events and commands:

- A *command* is a feature of the communications driver that can execute a particular function, such as on an external system like a telephone switch.

  Usually, the command, such as a request from the Siebel client to make a call or send a message, is executed with parameters whose values contain associated data, such as the number to call. The communications driver passes the command and associated data to the communications system for execution.

- An *event* is a notification of a communications occurrence that the Siebel client receives from the communications driver along with some data fields, such as notification of an incoming work item, with associated data such as a caller's telephone number (ANI).

For more information about interactive drivers, see "Interactive Drivers" on page 37.

Once you have written a communications driver, you configure Siebel Communications Server, as described in Chapter 4, "Configuring Siebel CTI," and Chapter 5, "Configuring Events and Commands."

Events and commands you define in the views in the Administration - Communications screen must be based on what the interactive communications driver supports. Commands, command parameters, events, and event data fields vary among communications drivers, just as they vary among vendors of communications systems, such as CTI middleware vendors.

You can retrieve driver parameter values from the Siebel database. Storing the parameters you need in the Siebel database allows the administrator to set parameter values appropriately for your communications driver.

You can write a communications driver to support the driver parameters you require. If you have created a custom driver to function in place of a driver provided with Siebel Business Applications, then you must then create all the driver parameters in the views in the Administration - Communications screen and assign values to them. If you have created a custom driver as an aggregate driver, to extend a Siebel driver, then you can use parameters from the Siebel driver.

The communications driver and the Siebel CRM software, such as Siebel business services and server components, work together to implement the communications behavior for the Siebel application.

Initialization of Communications Drivers

Driver initialization might involve initializing two handles for each communications driver implementation: ISC_DRIVER_HANDLE (the driver handle) and ISC_SERVICE_HANDLE (the service handle).

For drivers loaded on a Siebel Server computer, the driver handle is typically initialized only once, rather than once for each agent session. This depends on your driver implementation.

The driver handle and service handle can be implemented in any appropriate language, but must be implemented together using the same language.

Loading Communications Drivers

The process of loading a communications driver is as follows:

1. The channel manager invokes the driver handle method APIVersion, to check the driver's API version. The API version defined in the driver's header file must match the API version for Communications Server. For more information, see "APIVersion" on page 276.
2. The channel manager invokes the driver handle method CreateISCDriverInstance, to create the driver instance. The channel string passed to this method is collected from the driver record that is created in the Communications Drivers and Profiles view, in the Administration - Communications screen. For more information, see "CreateISCDriverInstance" on page 276.
3. The channel manager invokes the driver handle method RequestCommandEventList, to retrieve the list of events and commands this driver supports. For more information, see "RequestCommandEventList" on page 277.
4. The channel manager invokes the driver handle method RequestService, to create a service object. For more information, see "RequestService" on page 277.

If any listed step fails, then the driver loading fails. If multiple driver profiles must be loaded, then the process is repeated for each profile.

ISC_DRIVER_HANDLE

During initialization of an interactive communications driver, the CreateISCDriverInstance method is called to create the driver handle.

This method uses driver and profile parameter data (parameters prefaced with *Driver:* or parameters without a prefacing keyword) from the communications configuration, specified in the Communications Drivers and Profiles view.

For more information about the CreateISCDriverInstance method, see "CreateISCDriverInstance" on page 276.

ISC_SERVICE_HANDLE

After the driver handle has been created, the driver handle method RequestService is invoked, to request the driver handle to provide a service. Driver parameters (parameters prefaced with *Service:* or parameters without a prefacing keyword) are passed, which were specified in the Communications Drivers and Profiles view.

In response to this request, the driver handle returns the service handle (ISC_SERVICE_HANDLE), one instance for each agent. Calling methods of the service handle makes the feature set of the communications system available.

The driver handle and service handle together make up the implementation of the communications driver, along with other program data and logic.

Your implementation of the service handle invokes methods of the client handle, ISC_CLIENT_HANDLE. The client handle is implemented by Oracle.

For more information about the RequestService method, see "RequestService" on page 277.

Driver Parameters and Initialization

For the drivers provided with Siebel Business Applications, each driver parameter might be prefaced with a keyword indicating how it is used. Driver parameters defined for custom Adaptive Communications drivers can also follow this scheme for controlling how parameters are used:

- Parameters prefixed with *Driver:* are sent (with the prefix *Driver:* removed) to the driver handle when the driver is being initialized. These parameters are sent to the CreateISCDriverInstance method.
- Parameters prefaced with *Service:* are sent (with the prefix *Service:* removed) to the driver handle when it requests a service (creates the service handle). These parameters are sent to the RequestService method.
- Parameters that are not prefixed with either *Driver:* or *Service:* (as is the case for most of the drivers provided with Siebel Business Applications) are sent to the driver handle when the driver is being initialized (CreateISCDriverInstance method) and also when the driver handle requests a service (RequestService method).

Driver Event Attributes

Table 59 on page 266 lists event attributes that are included by the channel manager for all events processed by the client handle method HandleEvent. These attributes apply to any communications driver that invokes the client handle method HandleEvent, whether for drivers provided with Siebel Business Applications or custom drivers developed by customers or by other vendors.

You can use these attributes with event handler definitions in communications configurations. For more information, see "Event Handlers" on page 93 and "Work Item Attributes" on page 160.

The event attributes described in Table 59 are also available for workflows used in processing inbound communications.

For more information about the HandleEvent client handle method, see "HandleEvent" on page 274.

Table 59. Driver Event Attributes

Attribute Name	Description
SiebelChannelProfile	The name of the applicable profile for the driver.
SiebelChannelType	The language-independent value of the channel type.
SiebelChannelTypeString	The channel string defined for the driver.
SiebelDriverEventName	The device event name as known by the driver.

Table 59. Driver Event Attributes

Attribute Name	Description
SiebelDriverNotifyWhenDone	The setting of whether the driver requests notification from the client when event-handling is done. The client handle method HandleEvent includes the parameter notifyWhenDone, which is used to request the client to notify the driver when the client has finished handling the event. The client notifies the driver that event handling is done by using the service handle method NotifyEventHandlingFinished.
SiebelWorkItemID	The unique work item ID, a value in the form *profile_ID##driver_tracking_ID*, where *profile_ID* is the ID of the driver profile and *driver_tracking_ID* is the work item's tracking ID, as known by the driver. For example, if the ID of the driver profile is *12-01234*, and the driver uses the tracking ID *abcdefg* for this work item, then *12-01234##abcdefg* is the value of SiebelWorkItemID for this work item.

For events received for each interactive driver, additional data is received as event data fields. For more information about the events and event data fields for the Virtual CTI driver provided with Siebel Business Applications, see "Virtual CTI Driver Events" on page 253.

Siebel Adaptive Communications API Reference

This topic provides a reference for the methods supported by the Adaptive Communications application programming interface. Siebel Adaptive Communications is a programmable software layer between Siebel applications and the communications system.

For more information for developing a communications driver using Siebel Adaptive Communications, see "Adaptive Communications Design" on page 262.

This topic describes the client, driver, and service handles and the constants used by Adaptive Communications; and describes the methods for these handles.

This topic contains the following information:

- "Handles for Adaptive Communications" on page 268
- "Constants for Communications Drivers" on page 268
- "Data Types for Communications Drivers" on page 272
- "Methods of ISC_CLIENT_HANDLE" on page 273
- "Methods of ISC_DRIVER_HANDLE" on page 276
- "Methods of ISC_SERVICE_HANDLE" on page 278

Handles for Adaptive Communications

The Adaptive Communications layer includes a set of constants and the three handles described in the following paragraphs. The syntax with which each Adaptive Communications method is expressed is that of a C header file. In the following handle names, ISC stands for Interface for Siebel Communications:

- **ISC_CLIENT_HANDLE.** The client handle, which provides methods called by the communications driver. Oracle has implemented ISC_CLIENT_HANDLE and its methods.

  For descriptions of the client handle methods, see "Methods of ISC_CLIENT_HANDLE" on page 273.

- **ISC_DRIVER_HANDLE.** The driver handle, which is created when communications are initialized (for example, when the first user logs in). ISC_DRIVER_HANDLE then creates ISC_SERVICE_HANDLE, from which services are requested. Customers writing communications drivers must implement methods of the driver handle.

  For descriptions of the driver handle methods, see "Methods of ISC_DRIVER_HANDLE" on page 276.

- **ISC_SERVICE_HANDLE.** The service handle, which is created by ISC_DRIVER_HANDLE when communications are initialized (for example, when a user logs in). The Siebel client calls ISC_SERVICE_HANDLE methods in order to perform actual communications functions. Customers writing communications drivers must implement methods of the service handle.

  For more information about these handles, see "Adaptive Communications Design" on page 262.

  For descriptions of the service handle methods, see "Methods of ISC_SERVICE_HANDLE" on page 278.

Constants for Communications Drivers

Each of the following elements enumerates one or more constants. Values for all fields are stored when information about objects is passed back and forth between the Siebel application and the communications driver.

For example, object information is passed in this manner when the client handle method UpdateObjectInformation is invoked. For more information, see "UpdateObjectInformation" on page 275.

SCCommandFlag

Communications commands have dynamic or run-time status, consisting of the bit-flags enumerated by SCCommandFlag. Zero or more bit-flags can apply to a given command at any given time.

NOTE: In developing a driver, specify the names of the status flags, such as SC_CF_DISABLED, SC_CF_BLINKING, and so on, rather than the integer values.

```
enum SCCommandFlag
{
    SC_CF_NOTSUPPORTED  = 1,
    SC_CF_DISABLED      = 2,
    SC_CF_CHECKED       = 4,
    SC_CF_BLINKING      = 8,
    SC_CF_NOPARAMSOK    = 16,
    SC_CF_STRPARAMSOK   = 32
};
```

Each bit-flag that can be associated with a command is described as follows:

- **SC_CF_NOTSUPPORTED.** The command is not supported.
- **SC_CF_DISABLED.** The command is supported, but is disabled at this time.
- **SC_CF_CHECKED.** The command is in a checked state; for example, when an agent is in Not Ready mode, the command to set the Not Ready state is checked. This flag can be used to specify a toggle state for commands in the communications toolbar or in the Communications submenu of the Tools application-level menu.
- **SC_CF_BLINKING.** Blinking is enabled for communications toolbar buttons, such as those for the Accept Work Item and Resume Work Item commands.
- **SC_CF_NOPARAMSOK.** The command does not require any parameters to execute. For example, the commands for releasing a call and for toggling the Not Ready state do not require parameters.
- **SC_CF_STRPARAMSOK.** The command can be executed by providing a single unnamed string parameter (no named parameter is provided). The command for placing a call, for example, requires only a single parameter. Such commands are invoked when the user types something in the input field in the communications toolbar, such as an extension number, and clicks the command's button.

NOTE: If a device command's status excludes SC_CF_NOTSUPPORTED or SC_CF_DISABLED bit flags, then the command is considered enabled.

SCCommandTypeEx

Communications commands can be specified using either the InvokeCommand or InvokeCommandEx methods of the service handle. The values enumerated by SCCommandTypeEx specify the command type for an invocation of the InvokeCommandEx method. For more information, see "InvokeCommandEx" on page 279.

NOTE: In developing a driver, specify the names of the command types, such as SC_CT_MAKECALL, rather than the integer values.

```
    enum SCCommandTypeEx
    {
        SC_CT_MAKECALL        = 1,
        SC_CT_SENDMESSAGE     = 2
    };
```

Each command type is described as follows:

- **SC_CT_MAKECALL.** The command to be invoked is a command for placing a phone call. The InvokeCommandEx command includes predefined parameters concerning the phone call to be made.

- **SC_CT_SENDMESSAGE.** The command to be invoked is a command to send an email message. The InvokeCommandEx command includes predefined parameters concerning the email message to be sent.

SCErrorCode

The values enumerated by SCErrorCode specify predefined error codes. The error codes from 0 to 1000 are reserved.

NOTE: In developing a driver, specify the names of the error codes, such as SC_EC_OK, SC_EC_ERROR, and so on, rather than the integer values.

```
enum SCErrorCode
{
    SC_EC_OK                       = 0,
    SC_EC_ERROR                    = 1,
    SC_EC_CMD_NOT_SUPPORTED        = 2,
    SC_EC_MEDIA_TYPE_NOT_SUPPORTED = 3,
    SC_EC_INVALID_HANDLE           = 4,
    SC_EC_OUT_OF_MEMORY            = 5,
    SC_EC_NETWORK_ERROR            = 6,
    SC_EC_LIB_LOAD_ERR             = 7,
    SC_EC_FUNC_NOT_RESOLVED        = 8,
    SC_EC_DRIVER_CREATION_ERR      = 9,
    SC_EC_DRIVER_RELEASE_ERR       = 10,
    SC_EC_SERVICE_CREATION_ERR     = 11,
    SC_EC_SERVICE_RELEASE_ERR      = 12,
    SC_EC_INVALID_ITEM_TRACKING_ID = 13,
    SC_EC_CLIENT_INTERFACE_ERR     = 14,
    SC_EC_SENDMSG_FAILED_RETRY     = 15,
    SC_EC_IMPOBJ_CREATE_ERR        = 16,
    SC_EC_INVALID_LICENSE          = 17,
    SC_EC_WORK_ITEM_WRONG_STATE    = 18,
    SC_EC_DRIVER_SPECIFIC          = 1000
};
```

Each error code is described as follows:

- **SC_EC_OK.** The operation is successful.
- **SC_EC_ERROR.** The operation failed.
- **SC_EC_CMD_NOT_SUPPORTED.** The command is not supported.
- **SC_EC_MEDIA_TYPE_NOT_SUPPORTED.** The channel type string is not recognized.
- **SC_EC_INVALID_HANDLE.** Invalid handle.
- **SC_EC_OUT_OF_MEMORY.** Out of memory.
- **SC_EC_NETWORK_ERROR.** Networking error.
- **SC_EC_LIB_LOAD_ERR.** Failure on loading the driver file.
- **SC_EC_FUNC_NOT_RESOLVED.** Failure on resolving function address while invoking the command.

- **SC_EC_DRIVER_CREATION_ERR.** Failure on creating the driver handle.
- **SC_EC_DRIVER_RELEASE_ERR.** Failure on releasing the driver handle.
- **SC_EC_SERVICE_CREATION_ERR.** Failure on creating the service handle.
- **SC_EC_SERVICE_RELEASE_ERR.** Failure on releasing the service handle.
- **SC_EC_INVALID_ITEM_TRACKING_ID.** Invalid tracking ID.
- **SC_EC_CLIENT_INTERFACE_ERR.** Failure on invoking the ISC_CLIENT interface.
- **SC_EC_SENDMSG_FAILED_RETRY.** SC_CT_SENDMESSAGE failed, resend later.
- **SC_EC_IMPOBJ_CREATE_ERR.** Unable to create the underlying implementation object.
- **SC_EC_INVALID_LICENSE.** License error, used by the driver to report its license checking.
- **SC_EC_WORK_ITEM_WRONG_STATE.** The work item is in the wrong state for the operation.
- **SC_EC_DRIVER_SPECIFIC.** Specify for driver.

SCObjectProperty

Properties of monitored communications objects.

NOTE: In developing a driver, specify the names of the object properties, such as SC_OP_ONOFF, SC_OP_AGENTID, and so on, rather than the integer values.

```
enum SCObjectProperty
{
       SC_OP_ONOFF         = 1,
       SC_OP_AGENTID       = 2,
       SC_OP_ISNOTREADY    = 4,
       SC_OP_ISBUSY        = 5,
       SC_OP_DESCRIPTION   = 7,
       SC_OP_TIMEINQUEUE   = 9,
       SC_OP_QUEUEID       = 12,
       SC_OP_ISLOGON       = 13
};
```

Each object property is described as follows:

- **SC_OP_ONOFF.** A Boolean value indicating whether teleset control is on or off.
- **SC_OP_AGENTID.** The ID number of an agent, for an agent or DN that is being monitored.
- **SC_OP_ISNOTREADY.** A Boolean value indicating whether an agent has set the Not Ready state, for an agent or DN that is being monitored.
- **SC_OP_ISBUSY.** A Boolean value indicating whether an agent has set the Busy state, for an agent or DN that is being monitored.
- **SC_OP_DESCRIPTION.** A string describing the object that is being monitored.
- **SC_OP_TIMEINQUEUE.** The number of seconds during which the call stayed in the ACD queue before it was answered.

- **SC_OP_QUEUEID.** A string specifying the name of a queue. This is used together with SC_OP_ISLOGON to update the agent's login status for an ACD queue in the Communications options of the User Preferences screen.
- **SC_OP_ISLOGON.** A value indicating that the agent logged on or logged off.

SCWorkItemMode

Identifies work item modes, for inbound or outbound work items.

NOTE: In developing a driver, specify the names of the error codes, such as SC_WM_INBOUND, rather than the integer values.

```
enum SCWorkItemMode
{
      SC_WM_INBOUND  = 1,
      SC_WM_OUTBOUND = 2
};
```

Each work item type is described as follows:

- **SC_WM_INBOUND.** The work item is an inbound item.
- **SC_WM_OUTBOUND.** The work item is an outbound item, initiated by a user.

Data Types for Communications Drivers

The following are data types referenced in this appendix:

```
struct ISC_KeyValue       /* Key-Value element */
   {
      ISC_STRING    paramName;
      ISC_STRING    paramValue;
   };

struct ISC_KVParamList    /* List of Key-Value parameter */
   {
      struct ISC_KeyValue* dataItems;
      long                 len;
   };

struct ISC_StrParamList   /* List of String */
   {
      ISC_STRING*    dataItems;
      long           len;
   };

struct ISC_LongParamList  /* List of "long" */
   {
      long*          dataItems;
      long           len;
   };
```

Methods of ISC_CLIENT_HANDLE

The client handle, ISC_CLIENT_HANDLE, is implemented by Oracle. The communications driver calls client handle methods in order to communicate with the Siebel application, such as to send communications events and associated data.

When a call to a client handle method is successful, it returns 0 (zero). ISC_RESULT represents the API result type defined for the Siebel Adaptive Communications API.

BeginBatch

Begins a set of client methods to be invoked in a batch when the corresponding EndBatch method is invoked. Using these methods can reduce network overhead. Developers can use these methods at their discretion, subject to the driver implementation design. For more information, see "EndBatch" on page 273.

```
ISC_RESULT (*BeginBatch)
  /* in  */(ISC_CLIENT_HANDLE               handle);
```

CacheCommandInformation

Notifies the Siebel client about command-status caching.

For the *i*th command in commandNames, the command description is at the *i*th position in commandDescriptions, and the command status is at the *i*th position in commandStatuses. The SCCommandFlags bit flags are used by commandStatuses.

```
ISC_RESULT (*CacheCommandInformation)
  /* in  */(ISC_CLIENT_HANDLE               handle,
  /* in  */ const struct ISC_StrParamList*  commandNames,
  /* in  */ const struct ISC_StrParamList*   commandDescriptions,
  /* in  */ const struct ISC_LongParamList* commandStatuses);
```

CleanAllWorkItem

Notifies the Siebel client that all work items for this service session have been removed.

```
ISC_RESULT (*CleanAllWorkItem)
   /* in  */(ISC_CLIENT_HANDLE      handle);
```

EndBatch

Ends a set of client methods to be invoked in a batch when this method is invoked, given the use of the corresponding BeginBatch method to begin the batch. For more information, see "BeginBatch" on page 273.

```
ISC_RESULT (*BeginBatch)
  /* in  */(ISC_CLIENT_HANDLE               handle);
```

HandleError

Handles asynchronous errors:

- If clntCmdTrackID is not 0 (zero), then it contains the same value that was passed to the service handle method InvokeCommand that caused the error.
- If clntCmdTrackID is set to 0 (zero), then the error occurred out of context or the error was associated with a failed call attempt with which no call ID data is associated.

```
ISC_RESULT (*HandleError)
   /* in  */(ISC_CLIENT_HANDLE   handle,
   /* in  */ const ISC_STRING    clntCmdTrackID,
   /* in  */ const ISC_STRING    error);
```

HandleEvent

Handles the named event received from the communications driver, using the given fields. By calling this method, the communications driver notifies the Siebel client of a communications event, such as a call coming in on the monitored teleset.

If notifyWhenDone is set to True, then the communications driver is notified when event handling is finished. The event's ID is passed (using the trackingID parameter) in a call to the service handle method NotifyEventHandlingFinished. For more information, see "NotifyEventHandlingFinished" on page 279.

```
ISC_RESULT (*HandleEvent)
   /* in  */(ISC_CLIENT_HANDLE               handle,
   /* in  */ const ISC_STRING                name,
   /* in  */ const struct ISC_KVParamList*  fields,
   /* in  */ ISC_BOOLEAN                     notifyWhenDone,
   /* in  */ const ISC_STRING                trackingID);
```

IndicateNewWorkItem

Indicates a new incoming call by bringing the Siebel application to the front. If the end user has set the Enable Sound option in the User Preferences screen (Communications options), then the ringin.au file or another user-specified sound file plays.

```
ISC_RESULT (*IndicateNewWorkItem)
   /* in  */(ISC_CLIENT_HANDLE    handle,
   /* in  */ const ISC_STRING     trackingID,
   /* in  */ const ISC_STRING     oldTrackingID,
   /* in  */ const ISC_STRING     description,
   /* in  */ enum SCWorkItemMode  workItemMode);
```

ShowStatusText

Displays textual status information in the status line of the Siebel client.

```
ISC_RESULT (*ShowStatusText)
   /* in  */(ISC_CLIENT_HANDLE   handle,
   /* in  */ const ISC_STRING    text);
```

UpdateObjectInformation

Notifies the Siebel application about status changes of communications objects, for example, that an agent has become busy. Notifications about DN status are used to keep track of when calls end and new calls start. For the object properties that can be provided using datasetInfo, see "SCObjectProperty" on page 271.

```
ISC_RESULT (*UpdateObjectInformation)
   /* in  */(ISC_CLIENT_HANDLE              handle,
   /* in  */ const ISC_STRING               trackingID,
   /* in  */ const ISC_STRING               mediaTargetAddr,
   /* in  */ const struct ISC_KVParamList*  datasetInfo);
```

WorkItemReleased

After a work item has been released, such as a phone call disconnected, the driver calls this function.

```
ISC_RESULT (*WorkItemReleased)
   /* in  */(ISC_CLIENT_HANDLE     handle,
   /* in  */ const ISC_STRING      trackingID,
   /* in  */ time_t                stopTime);
```

WorkItemResumed

After a work item has been resumed, such as a phone call resumed from hold at the switch, the driver calls this function.

```
ISC_RESULT (*WorkItemResumed)
   /* in  */(ISC_CLIENT_HANDLE     handle,
   /* in  */ const ISC_STRING      trackingID);
```

WorkItemStarted

After a work item has been started, such as a phone call connected, the driver calls this function.

```
ISC_RESULT (*WorkItemStarted)
   /* in  */(ISC_CLIENT_HANDLE     handle,
   /* in  */ const ISC_STRING      trackingID,
   /* in  */ const ISC_STRING      oldTrackingID,
   /* in  */ const ISC_STRING      description,
   /* in  */ const ISC_STRING      mediaTargetAddr,
   /* in  */ time_t                startTime);
```

WorkItemSuspended

After a work item has been suspended, such as a phone call put on hold at the switch, the driver calls this function.

```
ISC_RESULT (*WorkItemSuspended)
   /* in  */(ISC_CLIENT_HANDLE     handle,
   /* in  */ const ISC_STRING      trackingID);
```

Methods of ISC_DRIVER_HANDLE

The driver handle, ISC_DRIVER_HANDLE, is implemented in the communications driver. Driver handle methods are invoked to request the driver to create a service handle, among other things. The service handle represents the functionality of the communications system with which you are integrating.

When a call to a driver handle method is successful, it returns 0 (zero). ISC_RESULT represents the API result type defined for the Siebel Adaptive Communications API.

APIVersion

Requests the driver to return the API version (Siebel Communications API, or SCAPI) for which the driver is implemented. The driver must return the SCAPI_VERSION defined in the header file.

```
ISCAPI long           APIVersion();
```

CreateISCDriverInstance

Requests the driver to create the driver handle ISC_DRIVER_HANDLE and to perform other initialization tasks, such as to connect to the CTI middleware.

```
ISCAPI ISC_RESULT   CreateISCDriverInstance
   /* in  */(const ISC_STRING              mediaTypeStr,
   /* in  */ const ISC_STRING              languageCode,
   /* in  */ const ISC_STRING              connectString,
   /* in  */ const struct ISC_KVParamList*  datasetParams,
   /* out */ ISC_DRIVER_HANDLE*              handle);
```

FreeSCStrParamList

Releases memory that was initially allocated for the driver.

```
ISCAPI void   FreeSCStrParamList
   /* in  */(struct ISC_StrParamList        strList);
```

NOTE: Each driver *must* implement this function, in order to free its own memory. If this function is not implemented, then the driver is not loaded.

GetImplementationObject

Returns a data structure for an implementation object, such as might be part of a communications driver implementation. This function might be useful for extending the functionality of an existing driver.

If your communications driver implementation aggregates another communications driver, such that you have an additional object that mediates communication with the driver handle and service handle, then this method can return implementation objects such as the driver handle, service handle, or some other structure.

An aggregate communications driver implementation typically includes a Siebel driver and a separate driver called the aggregate communications driver that extends the main driver's functionality or replaces part of its functionality.

If you write an aggregate communications driver, then it might be useful to invoke the aggregate driver for certain functions. The aggregate communications driver intercepts certain commands for its special operation and passes other commands on to the existing communications driver.

The aggregate driver must be added to the Communications Drivers and Profiles view, so that the driver can be properly initialized and used.

Writing an aggregate communications driver to work with an existing communications driver can save much time compared with writing a driver from scratch.

```
ISCAPI ISC_RESULT   GetImplementationObject
   /* in  */(ISC_HANDLE     key,
   /* out */ ISC_HANDLE*    impObj);
```

ReleaseISCDriverInstance

Releases the driver handle that was created using CreateISCDriverInstance, release a CTI middleware connection, and so on.

```
ISCAPI ISC_RESULT   ReleaseISCDriverInstance
   /* in  */(ISC_DRIVER_HANDLE             handle);
```

RequestCommandEventList

Returns the list of supported commands or events.

```
ISCAPI ISC_RESULT   RequestCommandEventList
   /* in  */(const ISC_STRING           mediaTypeStr,
   /* out */ struct ISC_StrParamList*    commandList,
   /* out */ struct ISC_StrParamList*    eventList);
```

RequestMediaTypeList

Returns the supported channel type.

```
ISCAPI ISC_RESULT   RequestMediaTypeList
   /* out */(struct ISC_StrParamList*     mediaTypeList);
```

RequestService

Requests the driver to create the service handle ISC_SERVICE_HANDLE and to pass the service type (such as teleset monitoring), user-defined parameters, and the pointer to the Siebel client handle. The driver handle creates the service handle, which in turn communicates with the Siebel application for each agent:

- For communications drivers provided with Siebel Business Applications, all parameter data is passed to the communications driver in the datasetParams parameter.

- For a customer-implemented communications driver, additional data can be passed to the service handle using the paramString variable.

```
ISCAPI ISC_RESULT   RequestService
   /* in  */(ISC_DRIVER_HANDLE                  handle,
   /* in  */ const struct ISC_CLIENT_INTERFACE  clntInterface,
   /* in  */ const ISC_STRING                   connectString,
   /* in  */ const struct ISC_KVParamList*      datasetParams,
   /* out */ ISC_SERVICE_HANDLE*                serviceHandle);
```

Methods of ISC_SERVICE_HANDLE

The service handle, ISC_SERVICE_HANDLE, is implemented in the communications driver. Services are returned to the Siebel client in response to a call for the driver handle method RequestService. The service handle methods are called in order to communicate with the communications system, for example, to send commands and associated data. For more information, see "RequestService" on page 277.

When a call to a service handle method is successful, it returns 0 (zero). ISC_RESULT represents the API result type defined for the Siebel Adaptive Communications API.

When a call to a service method fails, the communications driver can call the client handle method HandleError to pass error data. For more information, see "HandleError" on page 274.

AcceptWorkItem

Accepts a work item.

```
ISCAPI ISC_RESULT   AcceptWorkItem
   /* in  */(ISC_SERVICE_HANDLE       handle,
   /* in  */ const ISC_STRING         trackingID);
```

HandleQueuedWorkItem

Handles a queued work item.

```
ISCAPI ISC_RESULT   HandleQueuedWorkItem
   /* in  */(ISC_SERVICE_HANDLE             handle,
   /* in  */ const ISC_STRING               name,
   /* in  */ const struct ISC_KVParamList*  fields,
   /* in  */ const ISC_STRING               trackingID);
```

InvokeCommand

Invokes a command by name, using parameters. A basic ISC_SERVICE_HANDLE implementation must include at least the InvokeCommand method, in order to invoke named commands.

When this method is called, a command is invoked that is to be executed, for example, placing a call, transferring a call, or setting the Not Ready state. Most commands, but not all, are passed from the communications driver to the communications system.

```
ISCAPI ISC_RESULT   InvokeCommand
   /* in  */(ISC_SERVICE_HANDLE             handle,
   /* in  */ const ISC_STRING               clntCmdTrackID,
   /* in  */ const ISC_STRING               name,
   /* in  */ const ISC_STRING               stringParam,
   /* in  */ const struct ISC_KVParamList*  datasetParam);
```

InvokeCommandEx

Invokes a command by type, using parameters. The command type is represented by the value of the SCCommandTypeEx constant. You can use this method to invoke commands on an email server or CTI middleware server. The data structure ISC_KVParamList specifies the appropriate predefined parameters, according to the command type. For more information, see "SCCommandTypeEx" on page 269 and the description for InvokeCommand.

```
ISCAPI ISC_RESULT   InvokeCommandEx
   /* in  */(ISC_SERVICE_HANDLE             handle,
   /* in  */ const ISC_STRING               clntCmdTrackID,
   /* in  */ enum SCCommandTypeEx           commandType,
   /* in  */ const struct ISC_KVParamList*  datasetParam);
```

NotifyEventHandlingFinished

Notifies the communications driver that the Siebel client has finished handling the event that was sent by the communications driver to the Siebel client using a call for the client handle method HandleEvent. For more information, see "HandleEvent" on page 274.

```
ISCAPI ISC_RESULT   NotifyEventHandlingFinished
   /* in  */(ISC_SERVICE_HANDLE      handle,
   /* in  */ const ISC_STRING        trackingID,
   /* in  */ ISC_BOOLEAN             result);
```

ReleaseISCServiceInstance

Releases the service handle that was created using RequestService, and free resources. No subsequent calls to the driver handle are made. The driver must in turn make no further calls to the client handle.

NOTE: ReleaseISCServiceInstance is the last communication between the Communications Session Manager and the communications driver. After calling this function, Communications Session Manager no longer calls the driver, and the driver must no longer call back to the Siebel application, because the communications client and the client handle no longer exist.

```
ISCAPI ISC_RESULT   ReleaseISCServiceInstance
   /* in  */(ISC_SERVICE_HANDLE    handle);
```

ReleaseWorkItem

Releases a work item.

```
ISCAPI ISC_RESULT   ReleaseWorkItem
   /* in  */(ISC_SERVICE_HANDLE      handle,
   /* in  */ const ISC_STRING        trackingID);
```

ResumeWorkItem

Resumes a work item.

```
ISCAPI ISC_RESULT   ResumeWorkItem
   /* in  */(ISC_SERVICE_HANDLE      handle,
   /* in  */ const ISC_STRING        trackingID);
```

RevokeQueuedWorkItem

Revokes a queued work item.

```
ISCAPI ISC_RESULT     RevokeQueuedWorkItem
   /* in  */(ISC_SERVICE_HANDLE      handle,
   /* in  */ const ISC_STRING        trackingID);
```

SuspendWorkItem

Suspends a work item.

```
ISCAPI ISC_RESULT   SuspendWorkItem
   /* in  */(ISC_SERVICE_HANDLE      handle,
   /* in  */ const ISC_STRING        trackingID);
```

Testing Communications Drivers

The Communications Driver Test Engine (Test Engine) is a standalone tool that simulates parts of a Siebel environment for you to test the reliability and capacity of the communications driver that you develop.

The Test Engine does not test the Communications Server or your communications configuration elements. You can use the Test Engine to test your communications driver when connected to your CTI middleware by specifying appropriate driver parameters in the definition file (DEF) that you develop for testing your communications driver.

Before using the Test Engine, you configure settings for your communications driver. The Test Engine writes the results of the test to log files. You can specify the log file names by configuring the DEF file. For additional information about communications drivers, see Chapter 3, “Configuring Communications Drivers and Profiles.”

Table 60 describes the format that your DEF file must adhere to in order to simulate multiple calls.

Table 60. Format of Definition File

This section ...	Must contain ...
Driver Parameter	Driver parameters to connect to the CTI middleware.
Agent#n Where n = the agent's identifier	Parameters necessary to initialize agent login.
Job	Description of job to simulate.
Task	Description of tasks that comprise the job. Enter the device commands here that a communications driver supports to make sure that the commands and the communications driver function correctly.

The following procedure describes how you run the Test Engine tool to test a communications driver.

To test a communications driver

1. Copy your DEF file into the BIN subdirectory of your Siebel Server or Siebel client installation directory.
2. From a command line, navigate to the BIN directory identified in Step 1.
3. At the command prompt, execute the following command:

   On Windows:

   ```
   CommDriverTestEngine.exe definition_file language_code
   ```

   On UNIX:

   ```
   CommDriverTestEngine definition_file language_code
   ```

   For example, on Windows:

   ```
   CommDriverTestEngine.exe anExampleDefinitionFile.def ENU
   ```

   The Test Engine tests your communications driver and writes the output to the specified log files.

B Siebel Communications Server Business Services

This appendix provides information about business services for Siebel Communications Server, including applicable business services, methods, and arguments. It includes the following topics:

- About Business Services for Siebel Communications Server on page 283
- Communications Client Methods on page 283
- Communications Session Manager Methods on page 290

About Business Services for Siebel Communications Server

The following are the business services of Siebel Communications Server. The names shown are the display names for these business services:

- **Communications Client.** Supports the communications user interface features such as the communications toolbar and communications menu commands. This business service aggregates the Communications Session Manager business service. It does not deal with server-based communications functionality.
- **Communications Session Manager.** Provides an interface to session-based communications functionality at the server level. Through Server Request Broker and Server Request Processor, this business service communicates with the Communications Session Manager server component. It does not deal with the communications user interface.
- **Outbound Communications Manager.** Provides an interface to the outbound communications functionality of the Communications Outbound Manager server component. For more information about this business service, see *Siebel Email Administration Guide*.

For more information about integrating business services with Communications Server, see "Using Business Services with Siebel Communications Server" on page 181.

Communications Client Methods

This topic describes the methods and method arguments of the business service Communications Client. For more information about the role of business services and methods in communications toolbar configuration, see "About Communications Toolbar Configuration" on page 135.

The Communications Client business service is an aggregate business service that is built on top of the Communications Session Manager service. This means that you can invoke all the methods of the Communications Session Manager service as though they were methods of the Communications Client business service.

This topic primarily uses the display names for each method (Method Display Name). The internal names, which must be used in any scripts that invoke these methods, are also shown (Method Name).

Table 61 lists the methods for the Communications Client business service.

Table 61. Communications Client Methods

Method Display Name	Method Name	Comment
Agent Sign Off	AgentSignOff	Execute logout command
Agent Sign On	AgentSignOn	Execute login command
Get Selected Work Item Info	GetSelectedWorkItemInfo	Retrieve complete work item information of the selected work item on toolbar
Handle Error	HandleError	Push error information to the agent's communications toolbar
Is Comm Enabled	IsCommEnabled	Indicates whether communications are enabled
Make Call	MakeCall	Make a phone call
Obtain UI Focus	ObtainUIFocus	Bring the browser to the front (to gain user interface focus)
Reset Active Session Count	ResetActiveSessionCount	Reset the active session count.
Send Communication	SendCommunication	Send outbound communication item
Shell UI Update	ShellUIUpdate	Update or refresh the status of the communications toolbar
Show Status Text	ShowStatusText	Push error information to the agent's communications toolbar and display message on the browser's status bar
Work Item Released	WorkItemReleased	Notification of work item released
Work Item Resumed	WorkItemResumed	Notification of work item resumed
Work Item Started	WorkItemStarted	Notification of work item started
Work Item Suspended	WorkItemSuspended	Notification of work item suspended

Arguments for Communications Client Methods

This topic lists the arguments for each method of the business service Communications Client. Each argument's data type is also shown.

NOTE: In Siebel Tools, other arguments than these might also be listed. Disregard any arguments that are flagged as Hidden or Inactive.

Arguments for Agent Sign Off Method

The Agent Sign Off method has no arguments.

Arguments for Agent Sign On Method

The Agent Sign On method has no arguments.

Arguments for Get Selected Work Item Info Method

Table 62 lists the arguments for the Get Selected Work Item Info method.

Table 62. Arguments for Get Selected Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Type (type: String)	ChannelType	Output	Yes	The language-independent channel type
Channel Type Locale (type: String)	ChannelTypeLocale	Output	Yes	The language-dependent channel type locale
Description (type: String)	Description	Output	Yes	Work item description
Driver Work Tracking ID (type: String)	DriverWorkTrackID	Output	Yes	The tracking ID of this work item in driver scope
Icon File (type: String)	IconFile	Output	Yes	Icon file of the driver profile that owns this work item
Is Active State (type: Number)	IsActiveState	Output	Yes	Indicates if this work item is active
Is Data Contained Inside (type: Number)	HasWorkData	Output	Yes	Indicates if this work item contains data inside
Is Inbound Item (type: Number)	IsInboundItem	Output	Yes	Indicates if this is an inbound work item
Parent Work Item ID (type: String)	ParentWorkItemID	Output	Yes	ID of parent work item
Profile ID (type: String)	ProfileID	Output	Yes	ID of driver profile that owns this work item

Table 62. Arguments for Get Selected Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Profile Name (type: String)	ProfileName	Output	Yes	Name of driver profile that owns this work item
Time In Queue (type: String)	TimeInQueue	Output	Yes	Time this work item stayed in the queue
View Bookmark (type: String)	ViewBookmark	Output	Yes	View bookmark
Work Duration (type: String)	WorkDuration	Output	Yes	Duration of work item
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID
Work Item Key (type: String)	WorkItemKey	Output	Yes	Work item key
Work Item Not Exist (type: Number)	WorkItemNotExist	Output	Yes	Work item does not exist
Work Object ID (type: String)	WorkObjectID	Input	Yes	Object ID
Work Start Time (type: String)	WorkStartTime	Output	Yes	Time work item started
Work State (type: String)	WorkState	Output	Yes	Work item state
Work Tracking Object Business Component (type: String)	WorkTrackObjBusComp	Output	Yes	Name of business component for after-work tracking
Work Tracking Object Business Object (type: String)	WorkTrackObjBusObj	Output	Yes	Name of business object for after-work tracking

Arguments for Handle Error Method

Table 63 lists the arguments for the Handle Error method.

Table 63. Arguments for Handle Error Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Error (type: String)	Error	Input	Yes	Error text

Arguments for Is Comm Enabled Method

Table 64 lists the arguments for the Is Comm Enabled method.

Table 64. Arguments for Is Comm Enabled Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Is Comm Enabled (type: Number)	IsCommEnabled	Output	Yes	Indicates if the communications toolbar is enabled. 1 means True and 0 means False.

Arguments for Make Call Method

Table 65 lists the arguments for the Make Call method.

Table 65. Arguments for Make Call Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Apply Dialing Rule (type: Number)	ApplyDialingRule	Input	No	If dialing rule needs to be applied
Phone Number (type: String)	PhoneNumber	Input	Yes	Phone number
Profile Name (type: String)	ProfileName	Input	Yes	Name of driver profile

Arguments for Obtain UI Focus Method

The Obtain UI Focus method has no arguments.

Arguments for Send Communication Method

Table 66 lists the arguments for the Send Communication method.

Table 66. Arguments for Send Communication Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Type (type: String)	CommType	Input	Yes	Supported values: Wireless Email Fax Page
Method Argument (type: String)	Method Argument	Input	Yes	

Arguments for Shell UI Update Method

The Shell UI Update method has no arguments.

Arguments for Show Status Text Method

Table 67 lists the arguments for the Show Status Text method.

Table 67. Arguments for Show Status Text Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Text (type: String)	Text	Input	Yes	The text that is displayed at the browser's status bar

Arguments for Work Item Released Method

Table 68 lists the arguments for the Work Item Released method.

Table 68. Arguments for Work Item Released Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Stop Time (type: Number)	StopTime	Input	Yes	Time that work item is released
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Resumed Method

Table 69 lists the arguments for the Work Item Resumed method.

Table 69. Arguments for Work Item Resumed Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Started Method

Table 70 lists the arguments for the Work Item Started method.

Table 70. Arguments for Work Item Started Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Profile (type: String)	MediaProfile	Input	Yes	Name of driver profile that owns this work item
Channel Type (type: String)	MediaType	Input	Yes	Language-independent value of the channel type
Description (type: String)	Description	Input	Yes	Work item description
Object ID (type: String)	ObjectID	Input	Yes	Object ID of work item

Table 70. Arguments for Work Item Started Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Old Work Item ID (type: String)	OldWorkItemID	Input	Yes	Old work item ID
Start Time (type: Number)	StartTime	Input	Yes	Time work item is started
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Suspended Method

Table 71 lists the arguments for the Work Item Suspended method.

Table 71. Arguments for Work Item Suspended Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Communications Session Manager Methods

This topic describes the methods and method arguments of the business service Communications Session Manager.

This topic primarily uses the display names for each method (Method Display Name). The internal names, which must be used in any scripts that invoke these methods, are also shown (Method Name).

Table 72 lists the methods for the Communications Session Manager business service.

Table 72. Communications Session Manager Methods

Method Display Name	Method Name	Comment
Accept Work Item	AcceptWorkItem	Accept inbound work item
Get Agent Extension	GetAgentExtension	Retrieve agent's active extension number
Get All Work Item ID	GetAllWorkItemID	Retrieve all work item IDs

Table 72. Communications Session Manager Methods

Method Display Name	Method Name	Comment
Get Inbound Work Item Attributes	GetInboundWorkItemAttr	Retrieve inbound work item attribute
Get Inbound Work Item Info	GetInboundWorkItemInfo	Retrieve complete information of inbound work item
Get Top Active Work Item ID	GetTopActiveWorkItemID	Retrieve the ID of most recent active work item
Get Top Idle Work Item ID	GetTopIdleWorkItemID	Retrieve the ID of most recent idle work item
Get Work Item Attributes	GetWorkItemAttr	Retrieve work item attributes
Get Work Item Info	GetWorkItemInfo	Retrieve complete information of a work item
Get Work Item Track Info	GetWorkItemTrackInfo	Retrieve the tracking info of work item
Invoke Command	InvokeCommand	Invoke device command For more information, see related methods in "Siebel Adaptive Communications API Reference" on page 267.
Invoke Extended Command	InvokeCommandEx	Invoke predefined device command For more information, see related methods in "Siebel Adaptive Communications API Reference" on page 267.
Notify Event Handling Finished	NotifyEventHandlingFinished	Notify driver that event handling is over
Release Work Item	ReleaseWorkItem	Release work item
Release Work Item by Activity ID	ReleaseWorkItemEx	Release work item by activity record ID
Resume Work Item	ResumeWorkItem	Resume work item
Set Work Item Attributes	SetWorkItemAttr	Set work item attributes
Suspend Work Item	SuspendWorkItem	Suspend work item
Work Item Released	WorkItemReleased	Notification of work item released
Work Item Resumed	WorkItemResumed	Notification of work item resumed

Table 72. Communications Session Manager Methods

Method Display Name	Method Name	Comment
Work Item Started	WorkItemStarted	Notification of work item started
Work Item Suspended	WorkItemSuspended	Notification of work item suspended

Arguments for Communications Session Manager Methods

This topic lists the arguments for each method of the business service Communications Session Manager. Each argument’s data type is also shown.

NOTE: In Siebel Tools, other arguments than these might also be listed. Disregard any arguments that are flagged as Hidden or Inactive.

Arguments for Accept Work Item Method

Table 73 lists the arguments for the Accept Work Item method.

Table 73. Arguments for Accept Work Item Method

Argument Display Name And Data Type	Argument Name	Type	Req’d	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Get Agent Extension Method

Table 74 lists the arguments for the Get Agent Extension method.

Table 74. Arguments for Get Agent Extension Method

Argument Display Name And Data Type	Argument Name	Type	Req’d	Comment
Agent Login (type: String)	AgentLogin	Input	Yes	Agent’s login name
Destination DN (type: String)	DestinationDN	Output	Yes	Agent’s active extension

Arguments for Get All Work Item ID Method

Table 75 lists the arguments for the Get All Work Item ID method.

Table 75. Arguments for Get All Work Item ID Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
ActiveWorkItem_N (type: String)	ActiveWorkItem_N	Output	Yes	Attribute names are in ActiveWorkItem_? format, such as ActiveWorkItem_0, ActiveWorkItem_1
IdleWorkItem_N (type: String)	IdleWorkItem_N	Output	Yes	Attribute names are in IdleWorkItem_? format, such as IdleWorkItem_0, IdleWorkItem_1

Arguments for Get Inbound Work Item Attributes Method

Table 76 lists the arguments for the Get Inbound Work Item Attributes method.

NOTE: The Attribute Name and Attribute Value arguments are available for use, though they are not listed in Siebel Tools as arguments for this method.

Table 76. Arguments for Get Inbound Work Item Attributes Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Attribute Name (type: String)	AttrName	Input	No	Name of the attribute
Attribute Value (type: String)	AttrValue	Output	No	Value for the attribute
Channel Type (type: String)	ChannelType	Input	Yes	Name of language-independent channel type
Item Index (type: Number)	ItemIndex	Input	Yes	Index of inbound work item

Arguments for Get Inbound Work Item Info Method

Table 77 lists the arguments for the Get Inbound Work Item Info method.

Table 77. Arguments for Get Inbound Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Type (type: String)	ChannelType	Input	Yes	The language-independent channel type
Channel Type Locale (type: String)	MediaTypeLocale	Output	Yes	The language-dependent channel type locale
Description (type: String)	Description	Output	Yes	Work item description
Driver Work Tracking ID (type: String)	DriverWorkTrackID	Output	Yes	Work tracking ID in driver scope
Icon File (type: String)	IconFile	Output	Yes	Icon file of driver profile that owns this work item
Is Active State (type: Number)	IsActiveState	Output	Yes	Indicates if this work item is active
Is Data Contained Inside (type: Number)	HasWorkData	Output	No	Indicates if this work item contains data inside
Is Inbound Item (type: Number)	IsInboundItem	Output	Yes	Indicates if this is an inbound work item
Item Index (type: Number)	ItemIndex	Input	Yes	The index of work item
Parent Work Item ID (type: String)	ParentWorkItemID	Output	Yes	ID of parent work item
Profile ID (type: String)	ProfileID	Output	Yes	Row ID of driver profile
Profile Name (type: String)	ProfileName	Output	Yes	Name of driver profile
Time In Queue (type: Number)	TimeInQueue	Output	Yes	Time work item in queue

Table 77. Arguments for Get Inbound Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
View Bookmark (type: String)	ViewBookmark	Output	Yes	View bookmark
Work Duration (type: String)	WorkDuration	Output	Yes	Duration of work item
Work Item ID (type: String)	WorkItemID	Output	Yes	Work item ID
Work Item Not Exist (type: Number)	WorkItemNotExist	Output	Yes	If such work item exists
Work Object ID (type: String)	WorkObjectID	Output	Yes	Work object ID
Work Start Time (type: String)	WorkStartTime	Output	Yes	Time work item started
Work State (type: String)	WorkState	Output	Yes	Work item state
Work Tracking Object Business Component (type: String)	WorkTrackObjBusComp	Output	Yes	Name of business component for after-work tracking
Work Tracking Object Business Object (type: String)	WorkTrackObjBusObj	Output	Yes	Name of business object for after-work tracking

Arguments for Get Top Active Work Item ID Method

Table 78 lists the arguments for the Get Top Active Work Item ID method.

Table 78. Arguments for Get Top Active Work Item ID Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Output	No	Work item ID

Arguments for Get Top Idle Work Item ID Method

Table 79 lists the arguments for the Get Top Idle Work Item ID method.

Table 79. Arguments for Get Top Idle Work Item ID Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Output	No	Work item ID

Arguments for Get Work Item Attributes Method

Table 80 lists the arguments for the Get Work Item Attributes method.

NOTE: The Attribute Name and Attribute Value arguments are available for use, though they are not listed in Siebel Tools as arguments for this method.

Table 80. Arguments for Get Work Item Attributes Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Attribute Name (type: String)	AttrName	Input	No	Name of the attribute
Attribute Value (type: String)	AttrValue	Output	No	Value for the attribute
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Get Work Item Info Method

Table 81 lists the arguments for the Get Work Item Info method.

Table 81. Arguments for Get Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Type (type: String)	MediaType	Output	Yes	The language-independent channel type
Channel Type Locale (type: String)	MediaTypeLocale	Output	Yes	The language-dependent channel type locale

Table 81. Arguments for Get Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Description (type: String)	Description	Output	Yes	Work item description
Driver Work Tracking ID (type: String)	DriverWorkTrackID	Output	Yes	The tracking ID of this work item in driver scope
Icon File (type: String)	IconFile	Output	Yes	Icon file of driver profile that owns this work item
Is Active State (type: String)	IsActiveState	Output	Yes	Indicates if this work item is active
Is Data Contained Inside (type: Number)	HasWorkData	Output	No	Indicates if this work item contains data inside
Is Inbound Item (type: String)	IsInboundItem	Output	Yes	Indicates if this is an inbound work item
Parent Work Item ID (type: String)	ParentWorkItemID	Output	Yes	ID of parent work item
Profile ID (type: String)	ProfileID	Output	Yes	ID of driver profile that owns this work item
Profile Name (type: String)	ProfileName	Output	Yes	Name of driver profile which owns this work item
Time In Queue (type: String)	TimeInQueue	Output	Yes	Time this work item stays in the queue
View Bookmark (type: String)	ViewBookmark	Output	Yes	View bookmark
Work Duration (type: String)	WorkDuration	Output	Yes	Duration of work item
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID
Work Item Key (type: String)	WorkItemKey	Output	Yes	Work item key

Table 81. Arguments for Get Work Item Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item Not Exist (type: Number)	WorkItemNotExist	Output	No	Work item does not exist
Work Object ID (type: String)	WorkObjectID	Output	Yes	Object ID
Work Start Time (type: String)	WorkStartTime	Output	Yes	Time work item started
Work State (type: String)	WorkState	Output	Yes	Work item state
Work Tracking Object Business Component (type: String)	WorkTrackObjBusComp	Output	Yes	Name of business component for after-work tracking
Work Tracking Object Business Object (type: String)	WorkTrackObjBusObj	Output	Yes	Name of business object for after-work tracking

Arguments for Get Work Item Track Info Method

Table 82 lists the arguments for the Get Work Item Track Info method.

Table 82. Arguments for Get Work Item Track Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
After Work Tracking Object Business Component (type: String)	AfcBusComp	Output	Yes	Name of business component for after-work tracking
After Work Tracking Object Business Object (type: String)	AfcBusObj	Output	Yes	Name of business object for after-work tracking

Table 82. Arguments for Get Work Item Track Info Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
After Work Tracking Object Row ID (type: String)	AfcRowID	Output	Yes	Record ID for after-work tracking
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Invoke Command Method

Table 83 lists the arguments for the Invoke Command method.

Table 83. Arguments for Invoke Command Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Command Tracking ID (type: String)	CommandTrackingID	Input	Yes	Command ID
Data Set (type: Hierarchy)	DataSet	Input	Yes	Data parameters for driver
Device Command (type: String)	DeviceCommand	Input	Yes	Command to be invoked in driver
Driver Profile ID (type: String)	DriverProfileID	Input	Yes	Row ID of driver profile which is invoked
Profile Name (type: String)	ProfileName	Input	Yes	Name of driver profile which is invoked
String Parameter (type: String)	StringParam	Input	Yes	Single string data

Arguments for Invoke Extended Command Method

Table 84 lists the arguments for the Invoke Extended Command method.

Table 84. Arguments for Invoke Extended Command Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Command Tracking ID (type: String)	CommandTrackingID	Input	Yes	Command ID
Command Type (type: Number)	CommandType	Input	Yes	Command type
Data Set (type: Hierarchy)	DataSet	Input	Yes	Data parameters for driver
Driver Profile ID (type: String)	DriverProfileID	Input	Yes	Row ID of driver profile which is invoked
Profile Name (type: String)	ProfileName	Input	Yes	Name of driver profile which is invoked

Arguments for Notify Event Handling Finished Method

Table 85 lists the arguments for the Notify Event Handling Finished method.

Table 85. Arguments for Notify Event Handling Finished Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Result (type: Number)	Result	Input	Yes	Result code
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Release Work Item Method

Table 86 lists the arguments for the Release Work Item method.

Table 86. Arguments for Release Work Item Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Release Work Item by Activity ID Method

Table 87 lists the arguments for the Release Work Item by Activity ID method.

Table 87. Arguments for Release Work Item By Activity ID Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Activity ID (type: String)	ActivityID	Input	Yes	Row ID of activity record

Arguments for Resume Work Item Method

Table 88 lists the arguments for the Resume Work Item method.

Table 88. Arguments for Resume Work Item Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Set Work Item Attributes Method

Table 89 lists the arguments for the Set Work Item Attributes method.

NOTE: For this method, you can set values for one or more attributes for the work item, using key-value pairs rather than arguments with fixed names. (By contrast, the methods Get Inbound Work Item Attributes and Get Work Item Attributes use the Attribute Name and Attribute Value arguments to store the name of a single attribute to retrieve, and its value, respectively.)

Table 89. Arguments for Set Work Item Attributes Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Suspend Work Item Method

Table 90 lists the arguments for the Suspend Work Item method.

Table 90. Arguments for Suspend Work Item Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Released Method

Table 91 lists the arguments for the Work Item Released method.

Table 91. Arguments for Work Item Released Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Stop Time (type: Number)	StopTime	Input	Yes	Time that work item is released
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Resumed Method

Table 92 lists the arguments for the Work Item Resumed method.

Table 92. Arguments for Work Item Resumed Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Started Method

Table 93 lists the arguments for the Work Item Started method.

Table 93. Arguments for Work Item Started Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Channel Profile (type: Number)	MediaProfile	Input	Yes	Name of driver profile that owns this work item
Channel Type (type: Number)	MediaType	Input	Yes	Language-independent value of channel string
Description (type: String)	Description	Input	Yes	Work item description
Object ID (type: String)	ObjectID	Input	Yes	Object ID of work item
Old Work Item ID (type: String)	OldWorkItemID	Input	Yes	Old work item ID
Start Time (type: Number)	StartTime	Input	Yes	Time work item is started
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

Arguments for Work Item Suspended Method

Table 94 lists the arguments for the Work Item Suspended method.

Table 94. Arguments for Work Item Suspended Method

Argument Display Name And Data Type	Argument Name	Type	Req'd	Comment
Work Item ID (type: String)	WorkItemID	Input	Yes	Work item ID

C Views for Communications Administration

This appendix provides information about some of the views in the Administration - Communications screen. It includes the following topics:

Views for Defining Configurations, Drivers, Profiles, Agents, and Telesets

The following views are used to specify communications configurations, communications drivers and profiles, telesets and extensions, and agents:

- **All Configurations.** Create, modify, copy, or delete a record for a named communications configuration and define or associate related data such as parameters, profiles, agents, commands, and events. For more information, see "Creating or Modifying a Communications Configuration" on page 44.
- **Communications Drivers and Profiles.** Specify parameter values for communications drivers and define profiles to provide access to the driver and to specify override values for selected parameters. For more information, see Chapter 3, "Configuring Communications Drivers and Profiles."
- **Agent General Profile.** View or modify information about call or contact center communications users (contact-center agents and supervisors), including their ACD agent logins and passwords and associated ACD queues. Also associate an agent with one or more configurations or telesets. For more information, see "Specifying Agents" on page 55.
- **All Telesets.** Specify the telesets in your call center. For each teleset, specify its extensions (according to the switch you use). You can specify authorized agents for the teleset here or in the Agent General Profile view. For hoteling telesets, specify the hoteling computer. For more information, see "Specifying Telesets" on page 58.
- **All Extensions.** View information about all defined extensions and view or modify each extension's type and associated teleset. For more information, see "Specifying Telesets" on page 58.

Views for Defining Events

The following views are used to specify event handlers, event responses, and event logs:

- **All Event Logs.** Create or modify each event log, specify parameters and values, and associate the event log with a configuration.
- **All Event Responses.** Create or modify each event response, specify parameters and values, and associate the event response with a configuration. Also associate the event response with one or more event logs.
- **All Event Handlers.** Create or modify each event handler, specify parameters and parameter values, and associate the event handler with a configuration, and optionally with a profile. Also associate the event handler with an event response.

For more information, see "Defining Communications Events" on page 62.

Views for Defining Commands

The following views are used to specify commands and command data definitions:

- **All Command Data.** Create or modify each command data definition, specify parameters and values, and associate the command data definition with a configuration.
- **All Commands.** Create or modify each command, specify parameters and values, and associate the command with a configuration, and optionally with a profile. Also associate the command with a command data definition.

For more information, see "Defining Communications Commands" on page 67.

View for Exploring Configurations

The Configuration Explorer view provides an alternative way to view or specify some of the configuration elements. It allows you to view or specify communications configuration data. For more information, see "Viewing All Communications Configuration Data" on page 46.

Views for Generating Reports or Reviewing Run-time Status Data

The following views allow you to generate reports about communications activities, or display run-time status information about agents and communications work items:

- **Contact Center Telephony Analytics.** View information to help you analyze aspects of your contact center performance. For more information, see "Generating Contact Center Telephony Analytics Reports" on page 196.
- **All Active Agent Status.** View information about the communications activities of agents. For more information, see "Viewing Communications Status Data" on page 197.

- **All Channel Items.** View information about active work items for each supported channel type. For more information, see "Viewing Communications Status Data" on page 197.

View for Specifying Settings for Notifications (Message Broadcasts)

The following view allows you to specify settings for notifications (message broadcasts):

Message Broadcasts. Specify notification settings. For more information, see *Siebel Applications Administration Guide*.

Views for Siebel Email Response

For information about the views used for Siebel Email Response and related features, see *Siebel Email Administration Guide*.

View for Siebel Chat

For information about the views used for Oracle's Siebel Chat, see *Siebel Chat Guide*.

Index