G*plus* Adapter 8.0

for Siebel CRM

Developer's Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to www.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders.

The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact Genesys Customer Care. Before contacting Customer Care, please refer to the ***Genesys Care Support Guide for On-Premises*** for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesys.com

Document Version: 80gp_dev_slcrm_07-2015_v8.0.201.00

Table of Contents

List of Procedures

Preface

Welcome to the *G*plus *Adapter 8.0 for Siebel CRM Developer's Guide.* In general, this document addresses only the interactions of the Genesys G*plus* Adapter components with other Genesys systems and products. Developers who are using the Siebel application development tools and services to implement the G*plus* Adapter should look at the Siebel documentation set for more information.

The *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide* may contain information that is useful for developers who need to customize the G*plus* Adapter 8.0 for Siebel CRM. You should have ready access to this document, if only to understand the standard G*plus* Adapter configurations that may initially have been installed at your location.

In brief, you will find the following information in this guide:

- An overview of the Media Routing Component's interface and customization options.
- An overview of the iWD Routing Component's interface and customization options.

This document is valid only for the 8.0 release of this product.

Note: For versions of this document created for other releases of this product, visit the Genesys Documentation website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

This preface contains the following sections:

For information about the related resources and about the conventions that are used in this document, see the supplementary material starting on page 161.

About G*plus* Adapter 8.0 for Siebel CRM

The G*plus* Adapter 8.0 for Siebel CRM is a software solution that provides seamless integration between the Siebel CRM and Genesys 8.0 solutions. This combination brings together Siebel's leading software applications and Genesys' contact center solutions.

The G*plus* Adapter provides a single point of access to the contact information. The Adapter brings together multiple media and channels, and provides access to the power of the Siebel software, promoting better contact relationships overall.

Intended Audience

This guide is intended for developers who will customize the behavior of the G*plus* Adapter for Siebel CRM. The guide assumes that:

- You are familiar with concepts related to the Siebel Enterprise Application Integration (EAI) architecture.
- You have a basic understanding of computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- You have a good understanding of database systems, including the specific database system that your application uses.
- You have a basic understanding of network design and operation.
- You are familiar with the network configurations used in your enterprise's computing environment.
- You have a good knowledge of the Siebel application development environment, including Siebel Tools and Siebel Workflow.
- (If you will be modifying the style sheet file:) You understand `.XSL` syntax and file structure.

You should also be familiar with the following Genesys solutions:

- Framework
- Universal Routing
- Outbound Contact Solution
- eServices (formerly Multimedia)

Note: Refer to the *Genesys Interoperability Guide* for further information about the appropriate Genesys applications version numbers.

Usage Guidelines

The Genesys developer materials outlined in this document are intended to be used for the following purposes:

- Creation of contact-center agent desktop applications associated with the Genesys software implementations.
- Server-side integration between the Genesys software and the third-party software.
- Creation of a specialized client application specific to the customer needs.

The Genesys software functions available for development are clearly documented. No undocumented functionality is to be utilized without Genesys' express written consent.

Conditions of Use

The following Conditions of Use apply in all cases for developers employing the Genesys developer materials outlined in this document:

1. Possession of interface documentation does not imply a right to use by a third party. The Genesys conditions for use, as outlined below or in the *Genesys Developer Program Guide*, must be met.
2. This interface shall not be used unless the developer is a member in good standing of the Genesys Interacts program or has a valid Master Software License and Services Agreement with Genesys.
3. A developer shall not be entitled to use any licenses granted hereunder unless the developer's organization has met or obtained all prerequisite licensing and software as set out by Genesys.
4. A developer shall not be entitled to use any licenses granted hereunder if the developer's organization is delinquent in any payments or amounts owed to Genesys.
5. A developer shall not use the Genesys developer materials outlined in this document for any general application development purposes that are not associated with the above-mentioned intended purposes for the use of the Genesys developer materials outlined in this document.
6. A developer shall disclose the developer materials outlined in this document only to those employees who have a direct need to create, debug, and/or test one or more participant-specific objects and/or software files that access, communicate, or interoperate with the Genesys API.
7. The developed works and Genesys software running in conjunction with one another (hereinafter referred to together as the "integrated solutions") should not compromise the data integrity. For example, if both the Genesys software and the integrated solutions can modify the same data, then modifications by either product must not circumvent the other product's

data integrity rules. In addition, the integration should not cause duplicate copies of data to exist in both the participant and Genesys databases, unless it can be assured that the data modifications propagate all copies within the time required by the typical users.

8. The integrated solutions shall not compromise data or application security, access, or visibility restrictions that are enforced by either the Genesys software or the developed works.
9. The integrated solutions shall conform to the design and implementation guidelines and restrictions described in the *Genesys Developer Program Guide* and the Genesys software documentation. For example:
   a. The integration must use only published interfaces to access the Genesys data.
   b. The integration shall not modify data in the Genesys database tables directly using SQL.
   c. The integration shall not introduce the database triggers or stored procedures that operate on the Genesys database tables.

Any schema extension to the Genesys database tables must be carried out using the Genesys developer software through the documented methods and features.

The Genesys developer materials outlined in this document are not intended to be used for the creation of any product with a functionality comparable to any Genesys products, including products similar or substantially similar to Genesys's current general-availability, beta, and announced products.

Any attempt to use the Genesys developer materials outlined in this document or any Genesys developer software contrary to this clause shall be deemed a material breach with immediate termination of this addendum, and Genesys shall be entitled to seek to protect its interests, including but not limited to, preliminary and permanent injunctive relief, as well as money damages.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@genesys.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Customer Care if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Customer Care

If you have purchased support directly from Genesys, please contact Genesys Customer Care.

Before contacting Customer Care, please refer to the *Genesys Care Support Guide for On-Premises* for complete contact information and procedures.

Document Change History

This section lists content that is new or that has changed significantly since the first release of this document. The most recent changes appear first.

New in Document Version v8.0.201.00

This document has been re-issued to support the G*plus* Adapter for Siebel CRM release 8.0.2. The following topics have been added or changed since the previous release of the document:

- Table 4, "The CampSynch - Create Campaign Method Input Parameters," on page 24.
- "XML Schema 2" on page 26.
- Table 6, "The CampSynch - Merge Calling List Begin Method Input Parameters," on page 30.
- "XML Schema 4" on page 30.
- "XML Schema 5" on page 34.
- "XML Schema 10" on page 46.
- API version in all Examples of Chapter 1.
- Chapter "Campaign Synchronization Component 8.0.0" was removed.

New in Document Version v8.0.101.00

This document has been re-issued to support the G*plus* Adapter for Siebel CRM release 8.0.1. The following topics have been added or changed since the previous release of the document:

- The previously existing chapters, "The Campaign Synchronization Flow", "Configuring the List Import Functionality", "Using the Genesys Campaign Synchronization Business Service", and "Synchronization Summary Usage" have been combined under the following chapter, Chapter 1, "Campaign Synchronization Component 8.0.0," on page 15.
- The following chapter is now included in this document:
  - Chapter 1, "Campaign Synchronization Component," on page 15.

New in Document Version v8.0.002.00

- Chapter 1, “Campaign Synchronization Component 8.0.0,” on page 15 now includes the following new sections:
  - The Campaign Synchronization Data Flow.
  - Configuring the List Import Functionality.
  - The Genesys Campaign Synchronization Business Service.
  - The Default Scenarios for the Campaign Synchronization.
  - Synchronization Summary Usage.

Chapter

1 Campaign Synchronization Component

This chapter provides a developer's overview of the G*plus* Adapter Campaign Synchronization Component, focusing on the customization options and how to use them. The information contained herein is valid for the Campaign Synchronization Component release 8.0.1 and later. Each topic in this chapter describes one aspect of the Component that you can use to customize the campaign synchronization between the Siebel and Genesys environments. The following topics are included in this chapter:

For a functional overview of the complete G*plus* Adapter and its integration with the Siebel software, see the *G*plus *Adapter 8.0 for Siebel CRM User's Guide*. For a detailed description of the Siebel EAI architecture, see your Siebel documentation.

The Solution Overview

The 8.0.1 and later version of the G*plus* Adapter for Siebel CRM Campaign Synchronization Component is a new product in comparison to the earlier versions (7.5 and 8.0) of the Campaign Synchronization Component.

Campaign Synchronization Data Flow

Refer to the *G*plus *Adapter for Siebel CRM User's Guide* for the explanation of the data interaction between the Siebel and Genesys sections of the Adapter.

Configuring the List Import Functionality

Whereas your calls to the `Genesys Campaign Synchronization` business services' methods determine what synchronization tasks the Campaign Synchronization Component performs, the specific way in which the Campaign Synchronization Component maps the Siebel fields to the Genesys fields depends on two things:

- The Genesys Configuration Layer `Format` object that is specified by the `Format Application` configuration option. The `Format` object contains fields that form a structure for a database table.
- The custom fields and phone numbers specified during the deployment process.

The default value for the `Format` option used by the Campaign Synchronization Component is `GplusCampSynch`. You create the `Format` object and specify the custom fields and phone numbers according to the instructions provided in the *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide*.

Using the `Format` option, you can specify a different format at any time.

The Genesys Campaign Synchronization Server API Representation

The Campaign Synchronization Server provides a web-based interface to communicate with the clients. The Server acts as a simple web server that processes the HTTP messages of the POST request method.

Note: Other HTTP request methods (for example, GET) are not supported.

The input parameters of the Campaign Synchronization Server are transferred through the HTTP requests and the output parameters are transferred through the HTTP responses using XML messages.

The format of XML messages is specific to the application and is described in this document.

A single Campaign Synchronization Server can simultaneously process requests from several clients. Each client should be responsible for a unique set of campaigns and must have its own unique identifier. In the out-of-the-box solution, these clients are represented by the `CampSynch Configuration` Objects on Siebel side (see the *G*plus *Adapter for Siebel CRM User's Guide* for more details).

The Campaign Synchronization Server's API methods can be divided into the following three groups:

- The methods used to synchronize the campaign data from the Siebel side to the Genesys side:
  - `CampSynch - Create Campaign` (see, page 24)
  - `CampSynch - Delete Campaign` (see, page 27)
  - `CampSynch - Merge Calling List Begin` (see, page 29)
  - `CampSynch - Merge Calling List Data` (see, page 33)
  - `CampSynch - Merge Calling List Commit` (see, page 37)
  - `CampSynch - Delete Calling List` (see, page 39)
  - `CampSynch - Activate Calling List` (see, page 41)
  - `CampSynch - Deactivate Calling List` (see, page 43)
  - `CampSynch - Merge Contact Record` (see, page 45)
  - `CampSynch - Delete Contact Record` (see, page 49)
- The method used to query the updates (deltas) for the call results and number of call attempts:
  - `CampSynch - Query Call Results Delta` (see, page 55)
- The methods used to query the data from the Genesys side to the Siebel side (particularly, the query call results for an entire campaign, a calling list, or a contact.)
  - `CampSynch - Query Call Results by Campaign` (see, page 58)
  - `CampSynch - Query Call Results by Calling List` (see, page 59)
  - `CampSynch - Query Call Results by Contact` (see, page 61)

Each method group has its own specific behavior, parameters, and limitations—for example, only one request can be processed at a time for each of the first two method groups. This behavior is to ensure that the correct sequence of operations and the correct synchronization are performed as a result.

Frequently Used Input Parameters

There is a set of parameters that is commonly used for almost all of the requests. To avoid repetition, these parameters are collected in Table 1. They are referenced from the requests description.

Table 1: The Frequently Used Input Parameters

Parameter	Description
RequestId	The unique request identifier.
ConfigurationId	The client identifier. In the out-of-the-box solution, it is an identifier of a special configuration object in Siebel CRM that represents a connection to a particular Campaign Synchronization Server and is responsible for the synchronization of a particular set of campaigns. This parameter is used to avoid the simultaneous request execution in the scope of the aforementioned configuration object.
CampaignName	The campaign name to which a request is associated.
ListName	The calling list name to which a request is associated.
Summary	Specifies whether the information about the request execution process (synchronization summary) should be included into the output parameters (response). Valid values: `Y`: Include the synchronization summary. `N`: Do not include the synchronization summary.

Complex Type Parameters

There is a set of complex type parameters that are used in several of the requests. To avoid repetition, these parameters are collected in Table 2. They are referenced from the requests description.

Table 2: The Complex Type Parameters

Parameter	Sub-Parameter	Mandatory	Description
Field			The custom field description.
	FieldName	Yes	The data field name.
	FieldValue	Yes	The data field value.

Table 2: The Complex Type Parameters (Continued)

Parameter	Sub-Parameter	Mandatory	Description
Phone			The phone number.
	PhoneType	Yes	The phone type. This value must correspond to any Genesys phone type. Refer to the Outbound Contact Solution documentation for more details.
Phones			An array of phones. Can be set to zero-sized.
	Phone	No	A complex parameter. See, Phone.
CustomFields			An array of custom fields. Can be set ti zero-sized.
	Field	No	A complex parameter. See, Field.
ContactInfo			The Siebel contact data.
	crm_campaign_id	Yes	The Siebel campaign identifier.
	crm_contact_id	Yes	The global Siebel contact identifier.
	crm_camp_con_id	Yes	The Siebel contact identifier in scope of the particular Siebel campaign.
	tz_name	Yes	The contact's time zone name. Refer to the *G*plus *Adapter for Siebel CRM Deployment Guide* for more details on how to configure and synchronize the time zones between the Siebel side and the Genesys side.
	do_not_call	Yes	Specifies whether the contact should be marked as one that must not be called during campaign processing. Valid values: `Y`: Mark the contact as do not call. `N`: Do not mark the contact as do not call.
	Phones	Yes	A complex parameter. See, Phones.
	CustomFields	Yes	A complex parameter. See, CustomFields.

Table 2: The Complex Type Parameters (Continued)

Parameter	Sub-Parameter	Mandatory	Description
ContactRef			The reference to a Siebel contact
	crm_campaign_id	Yes	The Siebel campaign identifier.
	crm_contact_id	Yes	The global Siebel contact identifier.
	crm_camp_con_id	Yes	The Siebel contact identifier in scope of the particular Siebel campaign.
ContactList			An array of contacts. Can be set to zero-sized.
	ContactInfo	No	A complex parameter. See, ContactInfo.
CampaignList			An array of campaigns. Can be set to zero-sized.
	CampaignName	Yes	The campaign name to get the call results for.
	BatchMode	Yes	Specifies whether the batch mode for the campaign should be turned on or off. Valid values: `Y` - turn the batch mode on `N` - turn the batch mode off.

Synchronizing the Campaign Data from Siebel to Genesys

The methods listed in this group are designed to synchronize the campaign data from the Siebel side to the Genesys side.

These methods are designed to support the *double call tolerance* policy. It means that the same method can be called several times, but the result is the same—for example, if the required data object already exists then the corresponding creating method will do nothing but return a successful response. If the required object does not exist then the corresponding deleting method will do nothing but return a successful response.

The Method Descriptions

The entries in this section describe the following methods:

The Common Method's Response

All of the methods in this group produce responses of an identical type.

Table 3 describes the output parameters for this method.

Table 3: The Common Response Parameters

Parameter	Mandatory	Description
RequestId	Yes	The request identifier.
ResultCode	Yes	The result of the request execution process. This parameter can be one of following values: `100`: OK. Successfully executed. `200 - 299`: Confidential fault, which means that the request fails for some unrecoverable reason. `300 - 399`: Non-confidential fault, which means that a recoverable error occurs and the request can be successfully processed next time. `400` and higher: Critical fault, which means that not one request can be processed until the cause of the error is found.
ResultDescription	Yes	The textual result description.
Summary	No	The detailed information about the request execution process—the synchronization summary.

XML Schema 1 illustrates the XML schema of the common method response.

XML Schema 1

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ResultCode" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ResultDescription" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Summary" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 1 illustrates an example of the common method response.

Example 1

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<GCampSynchResponse APIVersion="2.2" Name="CampSynch - Merge Calling List Begin">
  <RequestId>1-1TSD</RequestId>
  <ResultCode>100</ResultCode>
  <ResultDescription>OK</ResultDescription>
  <Summary>
    Request: 'CampSynch - Merge Calling List Begin'
    Parameters:
    CampaignName = 'TestOL1_Load_1_ol'
    ListName = 'TestOL1_Load_1_Wave_1_ol'
    FolderName = ''
    InitialListState = 'Activated'
    Actions:
    [CME] Root folder 'Campaigns' found
    [CME] Root folder 'Campaigns' found
    [CME] Folder '/f1/f2' already exists
    [CME] Table access 'TestOL1_Load_1_Wave_1_ol' has been created
    [CME] Root folder 'Campaigns' found
    [CME] Folder '/f1/f2' already exists
    [CME] Table name 'CLT_123' has been updated
    [CME] Calling list 'TestOL1_Load_1_Wave_1_ol' has been created
    [CME] Calling list 'TestOL1_Load_1_Wave_1_ol' has been appended
    [CME] Calling list 'TestOL1_Load_1_Wave_1_ol' has been activated
    [DB] table: 'CLT_123_copy' has been dropped
    [DB] table: 'CLT_123' has been created
    [DB] table: 'CLT_123_copy' has been created
    [DB] index: 'CLT_123_copy_chain_idx' has been created
    [DB] request history table is available
  </Summary>
</GCampSynchResponse>
```

The CampSynch - Create Campaign Method

This method creates a `Campaign` object in the Genesys Configuration Database. This `Campaign` object is created without any calling lists.

Note: After a `Campaign` object is created and the calling lists are synchronized, you still must perform the manual steps to finish the campaign configuration by using Genesys Configuration Manager.

Table 4 describes the input parameters for the `CampSynch - Create Campaign` method.

Table 4: The CampSynch - Create Campaign Method Input Parameters

Parameter	Sub-Parameter	Mandatory	Description
RequestId		Yes	See, RequestId.
ConfigurationId		Yes	See, ConfigurationId.
CampaignName		Yes	See, CampaignName.
Summary		Yes	See, Summary.
GenesysTemplate Campaign		No	The Genesys campaign name that should be used as a template to finish the configuration. **Note:** This functionality is not implemented!

Table 4: The CampSynch - Create Campaign Method Input Parameters (Continued)

Parameter	Sub-Parameter	Mandatory	Description
FolderName		Yes	The folder name in the Configuration Database where a campaign is placed.
	type	Yes	Values can be `relative` or `absolute`. An absolute path specifies a full object path that starts from the tenant root (but does not include the tenant name). The absolute path enables using preconfigured `Configuration Units` and `Sites`. For example, an absolute path can be `Unit1/Campaigns/SBL` or `Site1/Unit2/Calling Lists/SBL/Org1`. If an absolute path is not specified, the relative path is used. A relative path specifies an object path that is relative to the default object path (`/Campaigns`, `/Calling Lists`, or `/Table Access`). For example, the `/SBL` relative path assumes the object creation under `/Campaigns/SBL`, `/Calling Lists/SBL`, and `/Table Access/SBL`. **Notes:** • An absolute path must contain a folder of type that corresponds to the required object type (for example, object type `Campaign` for campaigns, object type `Calling List` for calling lists, or object type `Table Access` for table access objects). • `Configuration Units` and `Sites` must be created prior to synchronization, along with a folder of the required type. All subfolders can be created automatically during synchronization.

XML Schema 2 illustrates the XML schema of the `CampSynch - Create Campaign` method request.

XML Schema 2

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="GenesysTemplateCampaign" type="xs:string" minOccurs="0"
              maxOccurs="1"/>
              <xs:element name="FolderName" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="type" type="xs:string" use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 2 illustrates a request to create a `Campaign` object named `TestOL1_Load_1` under the `Campaigns/folder_ol` folder. The synchronization summary should be included into the response.

Example 2

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Create Campaign">
  <NBParameters>
    <RequestId>1-20W5H</RequestId>
    <ConfigurationId>1-2A1B</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <Summary>Y</Summary>
    <FolderName type=relative>folder_ol</FolderName>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Delete Campaign Method

This method deletes a campaign from the Genesys side, including the objects found in the Configuration Database, the calling lists that are associated with the campaign, the Outbound Database tables that are related to the corresponding calling lists.

Table 5 describes the input parameters for the `CampSynch - Delete Campaign` method.

Table 5: The CampSynch - Delete Campaign Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
Summary	Yes	See, Summary.

XML Schema 3 illustrates the XML schema of the `CampSynch - Delete Campaign` method request.

XML Schema 3

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 3 illustrates a request to delete a campaign named `TestOL1_Load_1`. The synchronization summary should be included into the response.

Example 3

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Delete Campaign">
  <NBParameters>
    <RequestId>1-2DFP3</RequestId>
    <ConfigurationId>1-1Z1N1</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <Summary>Y</Summary>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Merge Calling List Begin Method

This method is the first of the three following methods listed that are designed for the calling list synchronization:

- `CampSynch - Merge Calling List Begin`
- `CampSynch - Merge Calling List Data`
- `CampSynch - Merge Calling List Commit`

These methods do the following merge operations under the calling list:

- If a calling list does not exist, then one is created.
- If a calling list exists, then the new contacts are merged with old ones.

This method creates the `Calling List` and `Table Access` objects in the Configuration Database, associates these objects, and then adds the calling list to the corresponding Genesys campaign.

The calling list can be added to the Genesys campaign in either the `active` or the `inactive` state. Whether or not the calling list is processed by the outbound solution depends on its state. The Genesys campaign to which the calling list should be added must be present in the Configuration Database at the moment of the execution request.

Table 6 describes the input parameters for the `CampSynch - Merge Calling List Begin` method.

Table 6: The CampSynch - Merge Calling List Begin Method Input Parameters

Parameter	Sub-Parameter	Mandatory	Description
RequestId		Yes	See, RequestId.
ConfigurationId		Yes	See, ConfigurationId.
CampaignName		Yes	See, CampaignName.
ListName		Yes	See, ListName.
Summary		Yes	See, Summary.
InitialListState		No	Specifies the state of the calling list. Valid values: 1. `Activated` (default) 2. `Deactivated`
CallingListFolderName		Yes	The folder name in the Configuration Database where a `Calling List` object is placed.
	type	Yes	See "The CampSynch - Create Campaign Method" on page 24 for more information.
TableAccessFolderName		Yes	The folder name in the Configuration Database where a `Table Access` object is placed.
	type	Yes	See "The CampSynch - Create Campaign Method" on page 24 for more information.

XML Schema 4 illustrates the XML schema of the `CampSynch - Merge Calling List Begin` method request:

XML Schema 4

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
```

```
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="CallingListFolderName" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="type" type="xs:string" use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="TableAccessFolderName" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="type" type="xs:string" use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="InitialListState" type="xs:string" minOccurs="0"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 4 illustrates a request to initiate the synchronization of the `TestOL1_Load_1_Wave_1` calling list to the `TestOL1_Load_1` campaign. The calling list should be activated. The synchronization summary should be included into the response.

Example 4

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Merge Calling List Begin">
  <NBParameters>
    <RequestId>1-1TSD</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
    <CallingListFolderName type="relative">/folder_ol</CallingListFolderName>
    <TableAccessFolderName type="relative">/folder_ol</TableAccessFolderName>
    <InitialListState>Activated</InitialListState>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Merge Calling List Data Method

This method is the second of the three following methods listed that are designed for the calling list synchronization:

- `CampSynch - Merge Calling List Begin`
- `CampSynch - Merge Calling List Data`
- `CampSynch - Merge Calling List Commit`

This method is designed to send a portion of the contacts data to the Genesys side. This method may be called a multiple of times, providing different portions of the contacts data each time.

Also, this method adds the corresponding Do Not Call contact's information into the Genesys Outbound Solution.

Table 7 describes the input parameters for the `CampSynch - Merge Calling List Data` method.

Table 7: The CampSynch - Merge Calling List Data Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.
ContactList	Yes	The list of the Siebel contacts. See the section, “Complex Type Parameters” on page 18.

XML Schema 5 illustrates the XML schema of the `CampSynch - Merge Calling List Data` method request:

XML Schema 5

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ContactList" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="ContactInfo">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="crm_campaign_id" type="xs:string" minOccurs="1"
        maxOccurs="1"/>
                          <xs:element name="crm_contact_id" type="xs:string" minOccurs="1"
        maxOccurs="1"/>
                          <xs:element name="crm_camp_con_id" type="xs:string" minOccurs="1"
        maxOccurs="1"/>
                          <xs:element name="tz_name" type="xs:string" minOccurs="1"
        maxOccurs="1"/>
                          <xs:element name="do_not_call" type="xs:string" minOccurs="1"
        maxOccurs="1"/>
                          <xs:element name="Phones">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="Phone">
                                  <xs:complexType>
```

```
                              <xs:simpleContent>
                                <xs:extension base="xs:string">
                                <xs:attribute name="PhoneType" type="xs:unsignedByte"
            use="required" />
                                <xs:attribute name="PhoneName" type="xs:string"
            use="required" />
                                </xs:extension>
                              </xs:simpleContent>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="CustomFields">
                      <xs:complexType>
                        <xs:sequence>
                         <xs:element name="Field" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="FieldName" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
                                <xs:element name="FieldValue" type="xs:string"
            minOccurs="1" maxOccurs="1"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 5 illustrates a request to pass three contacts of the TestOL1_Load_1_Wave_1 calling list of the TestOL1_Load_1 campaign. The synchronization summary is included into the response.

Example 5

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Merge Calling List Data">
  <NBParameters>
    <RequestId>1-1TSDTTTT1</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
    <ContactList>
      <ContactInfo>
        <crm_campaign_id>1-ERKI</crm_campaign_id>
        <crm_contact_id>1-103CF</crm_contact_id>
        <crm_camp_con_id>1-11BZI</crm_camp_con_id>
        <tz_name>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</tz_name>
        <do_not_call>N</do_not_call>
        <Phones>
          <Phone PhoneType="1" PhoneName="Calculated Home Phone #">4445557799</Phone>
          <Phone PhoneType="2" PhoneName="Calculated Work Phone #">4445557777</Phone>
        </Phones>
        <CustomFields></CustomFields>
      </ContactInfo>
      <ContactInfo>
        <crm_campaign_id>1-ERKI</crm_campaign_id>
        <crm_contact_id>1-11BZP123</crm_contact_id>
        <crm_camp_con_id>1-11K0Q</crm_camp_con_id>
        <tz_name>>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</tz_name>
        <do_not_call>N</do_not_call>
        <Phones>
          <Phone PhoneType="2" PhoneName="Calculated Work Phone #">3334445501</Phone>
        </Phones>
        <CustomFields></CustomFields>
      </ContactInfo>
      <ContactInfo>
        <crm_campaign_id>1-ERKI</crm_campaign_id>
        <crm_contact_id>1-11C02</crm_contact_id>
```

```
        <crm_camp_con_id>1-11K0T</crm_camp_con_id>
        <tz_name></tz_name>
        <do_not_call>Y</do_not_call>
        <Phones>
            <Phone PhoneType="1" PhoneName="Calculated Home Phone #">3334445502</Phone>
          </Phones>
        <CustomFields></CustomFields>
      </ContactInfo>
      </ContactList>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Merge Calling List Commit Method

This method is the third of the three following methods listed that are designed for the calling list synchronization:

- `CampSynch - Merge Calling List Begin`
- `CampSynch - Merge Calling List Data`
- `CampSynch - Merge Calling List Commit`

This method finalizes the synchronization process. It does the real contact data merging in the Outbound Database. Until this method is called, the Calling List data remains unchanged.

Note: If a calling list is processed by the Outbound Contact Solution while it is being synchronized, then some data collisions are possible. It is recommended to exclude the calling list from the Outbound processing during synchronization.

Table 8 describes the input parameters for the `CampSynch - Merge Calling List Commit` method.

Table 8: The CampSynch - Merge Calling List Commit Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.

XML Schema 6 illustrates the XML schema of the `CampSynch - Merge Calling List Commit` method request:

XML Schema 6

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
```

Example 6 illustrates a request to finalize the `TestOL1_Load_1_Wave_1` calling list of the `TestOL1_Load_1` campaign. The synchronization summary is included into the response.

Example 6

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Merge Calling List Commit">
  <NBParameters>
    <RequestId>1-1YTXK</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Delete Calling List Method

This method deletes a calling list from the Genesys side, including the `Table Access` object in the Configuration Database and the Outbound Database table that correspond to the calling list.

Table 9 describes the input parameters for the `CampSynch - Delete Calling List` method.

Table 9: The CampSynch - Delete Calling List Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.

XML Schema 7 illustrates the XML schema of the `CampSynch - Delete Calling List` method request:

XML Schema 7

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ConfigurationId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 7 illustrates a request to delete the `TestOL1_Load_1_Wave_1` calling list from the `TestOL1_Load_1` campaign. The synchronization summary is included into the response.

Example 7

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Delete Calling List">
  <NBParameters>
    <RequestId>1-1TS9</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Activate Calling List Method

This method activates a calling list on the Genesys side. The calling list must exist before this method is called. The calling list must be associated with the campaign specified in the request.

Table 10 describes the input parameters for the `CampSynch - Activate Calling List` method.

Table 10: The CampSynch - Activate Calling List Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.

XML Schema 8 illustrates the XML schema of the `CampSynch - Activate Calling List` method request:

XML Schema 8

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 8 illustrates a request to activate the `TestOL1_Load_1_Wave_1` calling list from the `TestOL1_Load_1` campaign. The synchronization summary is included into the response.

Example 8

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Activate Calling List">
  <NBParameters>
    <RequestId>1-1YTXO</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Deactivate Calling List Method

This method deactivates a calling list on the Genesys side. The calling list must exist before this method is called. The calling list must be associated with the campaign specified in the request.

Table 11 describes the input parameters for the `CampSynch - Deactivate Calling List` method.

Table 11: The CampSynch - Deactivate Calling List Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.

XML Schema 9 illustrates the XML schema of the `CampSynch - Deactivate Calling List` method request:

XML Schema 9

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 9 illustrates a request to deactivate the `TestOL1_Load_1_Wave_1` calling list from the `TestOL1_Load_1` campaign. The synchronization summary is included into the response.

Example 9

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Deactivate Calling List">
  <NBParameters>
    <RequestId>1-1YTXO</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Merge Contact Record Method

This method merges the specific contact information to a particular calling list on the Genesys side. If the requested contact information is not present in the calling list, it will be added. If the requested contact information is present, it will be updated. The contact record(s) is deleted, if the corresponding phone number is not present in the contact data. The calling list must exist before this method is called.

Table 12 describes the input parameters for the `CampSynch - Merge Contact Record` method.

Table 12: The CampSynch - Merge Contact Record Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.
ContactInfo	Yes	The Siebel contact data. See the section, “Complex Type Parameters” on page 18.

XML Schema 10 illustrates the XML schema of the `CampSynch - Merge Contact Record` method request:

XML Schema 10

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ContactInfo">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="crm_campaign_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_contact_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_camp_con_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="tz_name" />
                    <xs:element name="do_not_call" type="xs:string" />
                    <xs:element name="Phones" minOccurs="1" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="unbounded" name="Phone">
                            <xs:complexType>
                              <xs:simpleContent>
                                <xs:extension base="xs:unsignedLong">
```

```
                        <xs:attribute name="PhoneType" type="xs:unsignedByte"
            use="required" />
                        <xs:attribute name="PhoneName" type="xs:string"
            use="required" />
                      </xs:extension>
                    </xs:simpleContent>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="CustomFields" minOccurs="1" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Field" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="FieldName" type="xs:string" minOccurs="1"
            maxOccurs="1"/>
                      <xs:element name="FieldValue" type="xs:string" minOccurs="1"
            maxOccurs="1"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
  <xs:attribute name="Name" type="xs:string" use="required" />
 </xs:complexType>
</xs:element>
</xs:schema>
```

Example 10 illustrates a request to merge a contact of the `TestOL1_Load_1_Wave_1` calling list of the `TestOL1_Load_1` campaign. The synchronization summary is included into the response.

Example 10

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Merge Contact Record">
  <NBParameters>
    <RequestId>1-1TSDT</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
    <ContactInfo>
        <crm_campaign_id>1-ERKI</crm_campaign_id>
        <crm_contact_id>1-103CF</crm_contact_id>
        <crm_camp_con_id>1-11BZI</crm_camp_con_id>
        <tz_name>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</tz_name>
        <do_not_call>N</do_not_call>
        <Phones>
          <Phone PhoneType="2" PhoneName="Calculated Work Phone #">4445557777</Phone>
        </Phones>
        <CustomFields></CustomFields>
      </ContactInfo>
</Parameters>
</GCampSynchRequest>
```

The CampSynch - Delete Contact Record Method

This method deletes the specific contact information from a particular calling list on the Genesys side.

Table 13 describes the input parameters for the `CampSynch - Delete Contact Record` method.

Table 13: The CampSynch - Delete Contact Record Method Input Parameters

Parameter	Mandatory	Description
RequestId	Yes	See, RequestId.
ConfigurationId	Yes	See, ConfigurationId.
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
Summary	Yes	See, Summary.
ContactInfo	Yes	The Siebel contact data. See the section, "Complex Type Parameters" on page 18. **Note:** This contact data should not contain phones and custom fields as they are not required for this operation.

XML Schema 11on page 56 illustrates the XML schema of the `CampSynch - Delete Contact Record` method request:

XML Schema 11

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
       <xs:element name="RequestId" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Summary" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ContactInfo">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="crm_campaign_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_contact_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_camp_con_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="tz_name" />
                    <xs:element name="do_not_call" type="xs:string" />
                    <xs:element name="Phones" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="CustomFields" minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 11 illustrates a request to delete a contact of the TestOL1_Load_1_Wave_1 calling list of the TestOL1_Load_1 campaign. The synchronization summary is included into the response.

Example 11

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Delete Contact Record">
  <NBParameters>
    <RequestId>1-3G1RR</RequestId>
    <ConfigurationId>1-1F2X</ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <Summary>Y</Summary>
    <ContactInfo>
      <crm_campaign_id>1-2WQYQ</crm_campaign_id>
      <crm_contact_id>1-2WQRP</crm_contact_id>
      <crm_camp_con_id>1-2WRFN</crm_camp_con_id>
      <tz_name>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</tz_name>
      <do_not_call>N</do_not_call>
      <Phones></Phones>
      <CustomFields></CustomFields>
    </ContactInfo>
  </Parameters>
</GCampSynchRequest>
```

Querying the Updates (Deltas) for the Call Results and the Number of Call Attempts

The methods listed in this group are designed to query the data from the Genesys side to the Siebel side, in particular, to query the updates (deltas) for the call results and the number of call attempts for a campaign or a calling list.

The Method Descriptions

The entries in this section describe the following methods:

- The Common Method's Response, page 52
- The CampSynch - Query Call Results Delta Method, page 55
- The CampSynch - Query Call Results by Campaign Method, page 58
- The CampSynch - Query Call Results by Calling List Method, page 59
- The CampSynch - Query Call Results by Contact Method, page 61

The Common Method's Response

All of the methods in this group produce responses of an identical type.

Table 14 describes the output parameters for this method.

Table 14: The Common Response Parameters

Parameter	Mandatory	Description
ResultCode	Yes	The result of the request execution process. This parameter can be one of following values: `100`: OK. Successfully executed. `200 - 399`: Any kind of non-critical error. `400` and higher: Critical fault, which means that not one request can be processed until the cause of the error is found.
ResultDescription	Yes	The textual result description.
CallResultList	Yes	The list of the `CallResult` structures. Can be set to zero-sized. Each of the `CallResult` structures contain the following fields: • `crm_campaign_id`: The Siebel campaign identifier. • `crm_contact_id`: The global Siebel contact identifier. • `crm_camp_con_id`: The Siebel contact identifier in scope of the campaign. • `call_result`: The Genesys result code. • `call_time`: The time of the last call. • `attempts`: The call attempts number.

XML Schema 12 illustrates the XML schema of the common method response.

XML Schema 12

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ResultCode" type="xs:unsignedInt" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ResultDescription" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CallResultList" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="CallResult">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="crm_campaign_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_contact_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_camp_con_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="call_result" type="xs:unsignedByte" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="call_time" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="attempts" type="xs:unsignedByte" minOccurs="1"
              maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 12 illustrates an example of the common method response.

Example 12

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<GCampSynchResponse APIVersion="2.2" Name="CampSynch - Query Call Results by Campaign">
  <ResultCode>100</ResultCode>
  <ResultDescription>OK</ResultDescription>
  <CallResultList>
    <CallResult>
      <crm_campaign_id>1-2WQYQ</crm_campaign_id>
      <crm_contact_id>1-2WQS0</crm_contact_id>
      <crm_camp_con_id>1-2WRFO</crm_camp_con_id>
      <call_result>21</call_result>
      <call_time>1351004410</call_time>
      <attempts>1</attempts>
    </CallResult>
    <CallResult>
      <crm_campaign_id>1-2WQYQ</crm_campaign_id>
      <crm_contact_id>1-2WQSB</crm_contact_id>
      <crm_camp_con_id>1-2WRFP</crm_camp_con_id>
      <call_result>21</call_result>
      <call_time>1351004445</call_time>
      <attempts>1</attempts>
    </CallResult>
    <CallResult>
      <crm_campaign_id>1-2WQYQ</crm_campaign_id>
      <crm_contact_id>1-2WQS0</crm_contact_id>
      <crm_camp_con_id>1-2WRFO</crm_camp_con_id>
      <call_result>21</call_result>
      <call_time>1351004410</call_time>
      <attempts>2</attempts>
    </CallResult>
  </CallResultList>
</GCampSynchResponse>
```

The CampSynch - Query Call Results Delta Method

This method requests information about the call results and the number of call attempt that were collected since the previous method invocation (in other words, the call results delta).

The list of campaigns that are affected is provided as well as the method's input parameters. When a particular campaign is requested for the first time, the Campaign Synchronization Server starts collecting call results for this campaign. When a campaign disappears from the list, the Campaign Synchronization Server stops collecting the call results for it.

When a specific campaign is requested for the first time and the `batch mode` option is turned on for this campaign, the Campaign Synchronization Server queries the call results for the entire campaign and responds with this data.

Table 15 describes the input parameters for the `CampSynch - Query Call Results Delta` method.

Table 15: The CampSynch - Query Call Results Delta Method Input Parameters

Parameter	Mandatory	Description
ConfigurationId	Yes	See, ConfigurationId.
CampaignList	Yes	See, CampaignList.
InactivityTimeout	Yes	The specified timeout (in seconds) when the server should stop collecting the call results for a specific client (represented via the `ConfigurationId` parameter). When a client is not interested in anymore updates, it simply stops calling the request without notifying a server. So, this timeout prevents an endless loop of data collection.

XML Schema 13 illustrates the XML schema of the `CampSynch - Query Call Results Delta` method response.

XML Schema 13

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ConfigurationId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignList" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="CampaignName">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="BatchMode" type="xs:string" use="required" />
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="InactivityTimeout" type="xs: unsignedInt" minOccurs="1"
               maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Example 13 illustrates a request to obtain the call results delta for the `gpvmk3_campaign_1_Load_3`, `SiebelCamp1`, and `SiebelCamp2` campaigns.

XML Example 13

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Query Call Results Delta">
  <NBParameters>
    <ConfigurationId>1-1F2X </ConfigurationId>
  </NBParameters>
  <Parameters>
    <CampaignList>
      <CampaignName BatchMode="Y">gpvmk3_campaign_1_Load_3</CampaignName>
      <CampaignName BatchMode="N">SiebelCamp1</CampaignName>
      <CampaignName BatchMode="N">SiebelCamp2</CampaignName>
    </CampaignList>
    <InactivityTimeout>180</InactivityTimeout>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Query Call Results by Campaign Method

This method requests information about the call results and the number of call attempt for the entire Genesys campaign. Only the contact records that are already processed by the Outbound Contact Solution are included in this query.

Table 16 describes the input parameters for the `CampSynch - Query Call Results` method.

Table 16: The CampSynch - Query Call Results Method Input Parameters

Parameter	Mandatory	Description
CampaignName	Yes	See, CampaignName.

XML Schema 14 illustrates the XML schema of the `CampSynch - Query Call Results` method response.

XML Schema 14

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs ="1"/>
        <xs:element name="Parameters" minOccurs="1" maxOccurs ="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs
              ="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Example 14 illustrates a request to obtain the call results for the `TestOL1_Load_1` campaign.

XML Example 14

```
Request to obtain the call results for the 'TestOL1_Load_1'campaign.
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Query Call Results by Campaign">
  <NBParameters>
  </NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Query Call Results by Calling List Method

This method requests information about the call results and the number of call attempt for the entire Genesys calling list. Only the contact records that are already processed by the Outbound Contact Solution are included in this query.

Table 17 describes the input parameters for the `CampSynch - Query Call Results by Calling List` method.

Table 17: The CampSynch - Query Call Results by Calling List Method Input Parameters

Parameter	Mandatory	Description
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.

XML Schema 15 illustrates the XML schema of the `CampSynch - Query Call Results by Calling List` method response.

XML Schema 15

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
```

```
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
      <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Example 15 illustrates a request to obtain the call results for the `TestOL1_Load_1_Wave_1` calling list of the `TestOL1_Load_1` campaign.

XML Example 15

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Query Call Results by Calling List">
  <NBParameters></NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
  </Parameters>
</GCampSynchRequest>
```

The CampSynch - Query Call Results by Contact Method

This method requests information about the call results and the number of call attempt for a specific contact.

Table 18 describes the input parameters for the `CampSynch - Query Call Results by Contact` method.

Table 18: The CampSynch - Query Call Results by Contact Method Input Parameters

Parameter	Mandatory	Description
CampaignName	Yes	See, CampaignName.
ListName	Yes	See, ListName.
ContactRef	Yes	The Siebel contact reference. See the "Complex Type Parameters" on page 18.

XML Schema 16 illustrates the XML schema of the `CampSynch - Query Call Results by Contact` method response.

XML Schema 16

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="GCampSynchRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NBParameters" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Parameters" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CampaignName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ListName" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="ContactRef" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="crm_campaign_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_contact_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                    <xs:element name="crm_camp_con_id" type="xs:string" minOccurs="1"
              maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="APIVersion" type="xs:decimal" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XML Example 16 illustrates a request to obtain the call results for a specific contact of the `TestOL1_Load_1_Wave_1` calling list of the `TestOL1_Load_1` campaign.

XML Example 16

```
<?xml version="1.0" encoding="UTF-8"?>
<GCampSynchRequest APIVersion="2.2" Name="CampSynch - Query Call Results by Contact">
  <NBParameters></NBParameters>
  <Parameters>
    <CampaignName>TestOL1_Load_1</CampaignName>
    <ListName>TestOL1_Load_1_Wave_1</ListName>
    <ContactRef>
      <crm_campaign_id>1-ERKI</crm_campaign_id>
      <crm_contact_id>1-11BZP</crm_contact_id>
      <crm_camp_con_id>1-11K0Q</crm_camp_con_id>
    </ContactRef>
  </Parameters>
</GCampSynchRequest>
```

The Siebel Campaign Synchronization API Representation

The Siebel-side solution of the Campaign Synchronization Component 8.0.1and later consists of the following major parts:

- The tables and business components used to store the special data involved in the synchronization process.
- The applets and views used to configure and monitor the synchronization process.
- The business services that do almost all of the work required for the campaign synchronization.
- The workflow processes that are responsible for the unidirectional communication with the Campaign Synchronization Server.

Although, you can customize all of these parts, Genesys does not recommend modifying the tables and the business components because the entire solution is based on them and any changes may cause it not to work properly.

You can customize the applets and the views using the general Siebel procedures.

You can adjust the synchronization process to meet your needs by using the business services provided with the solution—for example, you can synchronize the campaigns/waves/contacts

Using the business services provided with the solution customers can adjust the synchronization process for their needs. For example, customers are able to synchronize campaigns/waves/contacts as a trigger to the Siebel runtime events that are not handled in the out-of-the-box solution. Also, you can customize the synchronization procedures from any appropriate place, such as menu items or buttons.

The Business Services Overview

The Campaign Synchronization Component solution provides six business services that are responsible for the different parts of the synchronization process.

For a brief description of the business services, see Table 19.

Table 19: The Campaign Synchronization Component Business Services

Business Service	Description
Genesys CampSynch Campaign	Provides the methods to generate requests for data synchronization. This business service generates requests and places them into the request queue, which are then read by the workflow process and then sent to the Campaign Synchronization server.
Genesys CampSynch Request Executor	Provides the method that reads the request from the queue and sends it to Genesys. This method is used by the workflow process.
Genesys CampSynch Call Result	Provides the methods to request and apply the call results from the Genesys side.
Genesys CampSynch Event Handler	Provides the methods that perform the event-oriented actions—for example, the reaction on the campaign loading or the contact record update.
Genesys CampSynch Utils	Provides the methods that perform the auxiliary actions. This method is intended to be called from the other business services.
Genesys CampSynch Tools	Provides the methods that are used during the solution deployment or the configuration procedure.

The Genesys CampSynch Campaign Business Service

The `Genesys CampSynch Campaign` business service provides the methods that generate the requests for the data synchronization from the Siebel side to the Genesys side. These requests are a part of the Siebel-side API for communication with the Campaign Synchronization Server.

Each of these methods generates an XML-based request, places it into a temporary file, and then adds an appropriate record into the request queue. These requests are then read from the request queue and then sent to the Campaign Synchronization Server (this process prevents any data loss in the event of a Campaign Synchronization Server failure). A path to the directory where the aforementioned temporary files are stored can be configured using the `FileSystem` user property of the business service (see the *G*plus *Adapter for Siebel CRM Deployment Guide* for more details).

Note: These methods do not directly send a request to the Campaign Synchronization Server.

The methods for the `Genesys CampSynch Campaign` business service can only be called for the Siebel campaigns that are properly configured for the synchronization process and that are enabled for the synchronization process.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Campaign` business service:

The CreateCampaign Method

This method generates a request to create a `Campaign` object on the Genesys side that corresponds to a particular Siebel campaign load.

Note: This method is not responsible for creating the calling lists that belong to the Siebel campaign load.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Create Campaign`

The DeleteCampaign Method

This method generates a request to delete a `Campaign` object on the Genesys side that corresponds to a particular Siebel campaign load. The presence of campaign on the Genesys side is not mandatory for a successful request execution.

Note: This method will complete delete a campaign, including all of the associated calling lists. You do not have to call the `DeleteWave` method for Campaign Load Waves before using this method.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Delete Campaign`

The ExportWave Method

This method generates multiple requests to create or merge a calling list to the Genesys side that corresponds to a particular Siebel campaign load wave.

All contacts assigned to the wave are separated by pages (a relatively small number of contacts) and each page is then attached to its own request, which allows the synchronization process to be done more effectively in case of network failures. The page size can be set using the `ContactPageSize` user

property of the business service (see the *G*plus *Adapter for Siebel CRM Deployment Guide* for more details).

Note: An appropriate `CreateCampaign` method must be called before calling this method. Otherwise, because this method only puts the successfully processed requests into the queue, the execution of future requests will fail due to the absence of a campaign on the Genesys side.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.
`InitialState`	Optional: The initial state of a calling list on the Genesys side. Possible values are `Activated` and `Deactivated`. If the value is not set, then it is not set in the `CampSynch - Merge Calling List Begin` request and the request use the default behavior.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Merge Calling List Begin`

`CampSynch - Merge Calling List Data` (from `0`-zero to `N` times)

`CampSynch - Merge Calling List Commit`

The DeleteWave Method

This method generates a request to delete a calling list from the Genesys side that corresponds to a particular Siebel campaign load wave. The calling list presence on the Genesys side is not mandatory for a successful request execution.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Delete Calling List`

The StartWaveProcessing Method

This method generates a request to activate the processing of a calling list on the Genesys side that corresponds to a particular Siebel campaign load wave. The campaign and the calling list presence on the Genesys side is mandatory for a successful request execution.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Activate Calling List`

The StopWaveProcessing Method

This method generates a request to deactivate the processing of a calling list on the Genesys side that corresponds to a particular Siebel campaign load wave. The campaign and the calling list presence on the Genesys side is mandatory for a successful request execution.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Deactivate Calling List`

The ExportWaveRecord Method

This method generates a request to merge the contact list record on the Genesys side that corresponds to a particular record in the Siebel campaign load wave. The campaign and the calling list presence on the Genesys side is mandatory for a successful request execution.

Input Arguments

`CampContId`	Mandatory: The ID of the Siebel contact that belongs to a particular wave. The value must correspond to the `Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Merge Contact Record`

The DeleteWaveRecord Method

This method generates a request to delete the contact list record on the Genesys side that corresponds to a particular record in the Siebel campaign load wave. The campaign and the calling list presence on the Genesys side is mandatory for a successful request execution. The contact list record presence on the Genesys side is not mandatory.

Note: This method must be called until an appropriate record is present in the `Campaign List Contact` business component.

Input Arguments

`CampContId`	Mandatory: The ID of the Siebel contact that belongs to a particular wave. The value must correspond to the `Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Delete Contact Record`

The Genesys CampSynch Request Executor Business Service

The `Genesys CampSynch Request Executor` business service provides a single method that reads and executes (sends the request to the Campaign Synchronization Server and processes the response) one request from the request queue. This method is specially designed to be called from the Campaign Synchronization Component's workflow that is responsible for the request queue processing.

It is very important that for each particular `CampSynch Configuration` object (in other words, for each particular Campaign Synchronization Server) there must be only one process that executes requests. In the out-of-the-box solution, this process is the workflow process. If several processes execute requests for a particular Campaign Synchronization Server, it may cause an incorrect sequence execution and therefore a synchronization failure.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Request Executor` business service:

- The Execute Method, page 71

The Execute Method

This method reads from the request queue and executes one request, which means it sends the request to the appropriate Campaign Synchronization Server and processes the response.

The requests are chosen according to the execution order that is configured for each `CampSynch Configuration` object. See the *G*plus *for Siebel CRM User's Guide* for details about the order of the request execution.

This method is designed to be run from the out-of-the-box workflow process, therefore its input and output arguments are correlated with it.

Input Arguments

`CasConfObjId`	Mandatory: The ID of the `CampSynch Configuration` object that determines a synchronization target. This

	value must correspond to the `Id` field of the `Genesys CaS Config Object` business component.
`CampaignId`	Optional: The ID of the Siebel campaign for the previously executed request. Used for choosing the next request for a number of types of order execution. If this argument is provided, the value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component. The value for this argument can be obtained from the identically named output argument.
`ExecChainId`	Optional: The ID of a chain of the previously executed request. Used for choosing the next request in a chain (the requests can be chained in the queue when they perform a common business function—for example, requests for a wave/calling list synchronization). The value must be taken from the `Chain Id` field of the `Genesys CaS Synch Queue` business component. The value for this argument can be obtained from the identically named output argument.
`LastConnSubsystem`	Optional: The type (primary or backup) of the last successfully used connection subsystem (the Siebel profile configuration). Possible values are `Primary` and `Backup`. Used for choosing a connection subsystem to send a request first. If not provided, the primary connection subsystem is used first. The value for this argument can be obtained from the identically named output argument.

Output Argument

`ResultCode` The result code of a request execution. See Table 20 below for details:

Table 20: The Result Codes for a Request Execution (Execute)

Code	Result
0	The request is found and processed either successfully or with an unrecoverable error, which means that it is possible to try to execute the next request. If the request executed successfully, then the corresponding record is removed from the queue and the corresponding file (with an XML-based request content) is removed from the file system. If the request is executed with an unrecoverable error, then the corresponding record is marked as erroneous and it is kept in the queue for further analyzing and manual removal.
1	No request is found in the queue.
2	The request is found, but executed with a non-critical (recoverable) error—for example, a server outage, which means that it is possible to retry this request again. The corresponding record is not removed from the queue and the next method call will find this record again.
3	A critical error occurs—for example, an incorrect configuration is found or an unexpected exception occurred, which means that there is no reason to run the method again if the cause for the error is not corrected.
4	The signal to exit from the workflow process. At this time, the current implementation does not generate this code.

`ResultDescription` A more detailed description of the results, which can be obtained from the Campaign Synchronization Server or generated by the implementation of the method itself.

`CampaignId` The ID of the Siebel campaign of the recently executed request. Used for choosing the next request for some types of request processing.

`ExecChainId` The ID of a chain of the recently executed request. Used for choosing the next request in a chain (requests can be chained in the queue when they do perform a common business function—for example, requests for a wave/calling list synchronization).

`LastConnSubsystem`	The type (primary or backup) of the last successfully used connection subsystem (the Siebel profile configuration). Possible values are `Primary` and `Backup`. Used for choosing a connection subsystem to send a request first.
`NoDataTimeout`	The value of the timeout, specified in seconds, that is configured (in the `Camp WF NoData Timeout` field of the `Genesys CaS Config Object` business component) for a particular `CampSynch Configuration` object. It is used by the out-of-the-box workflow process to make a pause when this method returns a result code value of `1`.
`RetryTimeout`	The value of the timeout, specified in seconds, that is configured (in the `Camp WF Retry Timeout` field of the `Genesys CaS Config Object` business component) for a particular `CampSynch Configuration` object. It is used by the out-of-the-box workflow process to make a pause when this method returns a result code value of `2`.

Exceptions

This method does not raise an exception.

The Genesys CampSynch Call Result Business Service

The `Genesys CampSynch Call Result` business service provides methods that generate requests for the data (call results) synchronization from the Genesys side to the Siebel side. These requests are part of the Siebel-side API for communication with the Campaign Synchronization Server.

Each of these methods generates an XML-based request, directly sends it to the appropriate Campaign Synchronization Server and then updates the Siebel Campaign Contacts with the received call results and the number of attempted calls.

The methods for the `Genesys CampSynch Campaign` business service can only be called for the Siebel campaigns that are properly configured for the synchronization process and that are enabled for the synchronization process.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Call Result` business service:

- The ImportCallResultsDelta Method, page 75
- The ImportWaveCallResults Method, page 77
- The ImportContactCallResults Method, page 78

The ImportCallResultsDelta Method

This method generates and sends a request for getting the call results delta, which is the call results for the contacts that were processed by the Genesys outbound solution since sending the previous request. Also, it applies the received call results and the number of call attempt to the corresponding Siebel campaign contacts. It is targeted to a particular Campaign Synchronization Server (through the `CampSynch Configuration` object) and requests the information for the campaigns that are linked to it. This method uses the configured call result synchronization settings, including the batch mode flag.

This method is designed to be run from the out-of-the-box workflow process, therefore its input and output arguments are correlated with it.

Input Arguments

`CasConfObjId` — Mandatory: The ID of the `CampSynch Configuration` object that determines a synchronization target. This value must correspond to the `Id` field of the `Genesys CaS Config Object` business component.

`LastConnSubsystem` — Optional: The type (primary or backup) of the last successfully used connection subsystem (the Siebel profile configuration). Possible values are `Primary` and `Backup`. Used for choosing a connection subsystem to send a request first. If not provided, the primary connection subsystem is used first. The value for this argument can be obtained from the identically named output argument.

Output Argument

`ResultCode` — The result code of a request execution. See Table 21 below for details:

Table 21: The Result Codes for a Request Execution (ImportCallResultsDelta)

Code	Result
`0`	The request is processed either successfully or with an unrecoverable error, which means that the next time a request should be sent is after the `QueryTimeout` delay.
1	The request is processed with a non-critical (recoverable) error—for example, a server outage, which means that the next request should be sent after the `RetryTimeout` delay.

Table 21: The Result Codes for a Request Execution (ImportCallResultsDelta) (Continued)

Code	Result
2	A critical error occurs—for example, an incorrect configuration is found or an unexpected exception occurred, which means that there is no reason to run the method if the cause of the error is not corrected.
3	The signal to exit from the workflow process. At this time, the current implementation does not generate this code.

`ResultDescription` A more detailed description of the results, which can be obtained from the Campaign Synchronization Server or generated by the implementation of the method itself.

`LastConnSubsystem` The type (primary or backup) of the last successfully used connection subsystem (the Siebel profile configuration). Possible values are `Primary` and `Backup`. Used for choosing a connection subsystem to send a request first.

`QueryTimeout` The value of the timeout, specified in seconds, that is configured (in the `CR WF Query Timeout` field of the `Genesys CaS Config Object` business component) for a particular `CampSynch Configuration` object. It is used by the out-of-the-box workflow process to make a pause when this method returns a result code value of `0` (zero).

`RetryTimeout` The value of the timeout, specified in seconds, that is configured (in the `CR WF Retry Timeout` field of the `Genesys CaS Config Object` business component) for a particular `CampSynch Configuration` object. It is used by the out-of-the-box workflow process to make a pause when this method returns a result code value of `1`.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Query Call Results Delta`

The ImportLoadCallResults Method

This method generates and sends request for getting the call results for an entire Siebel campaign load. Also, it applies the received call results and the number of call attempts to the corresponding campaign contacts.

Within the out-of-the-box solution, this method is called by the corresponding button on the GUI.

Note: The calling of this method may produce a large amount of data that is received in response to the request and is applied to the Siebel campaign contacts. Therefore, the request's execution may take a while.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Query Call Results by Campaign`

The ImportWaveCallResults Method

This method generates and sends request for getting the call results for an entire Siebel campaign load wave. Also, it applies the received call results and the number of call attempts to the corresponding campaign contacts.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.

`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Query Call Results by Calling List`

The ImportContactCallResults Method

This method generates and sends a request for getting the call results for a single particular Siebel campaign contact. Also, it applies the received call results and the number of call attempts to the corresponding campaign contacts.

Within the out-of-the-box solution, this method is not called, but can be applied during the customization, if required.

Input Arguments

`CampContId`	Mandatory: The ID of the Siebel contact that belongs to a particular wave. The value must correspond to the `Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

Generated CampSynch Server API Requests

`CampSynch - Merge Contact Record`

The Genesys CampSynch Event Handler Business Service

The `Genesys CampSynch Event Handler` business service implements reactions on different events related to the Siebel campaign life-cycle—for example, Campaign Loaded, Campaign Launched, Contact Changed.

These methods implement algorithms that determine and generate (using the `Genesys CampSynch Campaign` business service) requests to Campaign Synchronization Server depending on the following:

- The event that happened
- The Siebel Campaign/Wave state
- The Campaign synchronization mode (configured using the GUI for each campaign)

All of the methods of the `Genesys CampSynch Event Handler` business service can only be called for the Siebel campaigns that are properly configured for the synchronization process and that are enabled for the synchronization process. This method does not have any effect on any of the other campaigns.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Event Handler` business service:

The CampLoadWave WriteRecord Method

This method implements a reaction to the event of writing a `Campaign Load Wave` business component record. It determines the following:

- Whether the campaign is loaded or purged and calls an appropriate following method—`The OnCampaignLoaded Method`, `The OnCampaignPurged Method`, `The OnWaveLaunched Method`, `The OnWaveSuspended Method`, or `The OnWaveUpdated Method`.

- Whether the wave is launched, suspended or updated and calls an appropriate following method—The OnCampaignLoaded Method, The OnCampaignPurged Method, The OnWaveLaunched Method, The OnWaveSuspended Method, or The OnWaveUpdated Method.

This method is designed to be called by the out-of-the-box Siebel runtime event handlers. It uses the Profile attributes to store the input and output data. These Profile attributes are set by the runtime event handlers when the appropriate business component's fields are changed.

Input Arguments

None.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The Contact WriteRecord Method

This method implements a reaction to the event of writing Contact-containing business components—for example, Contact, Employee, and others. It determines whether the contact is updated and calls the The OnContactUpdated Method when required.

This method is designed to be called by the out-of-the-box Siebel runtime event handlers. It uses the Profile attributes to store the input and output data. These Profile attributes are set by the runtime event handlers when the appropriate business component's fields are changed.

Input Arguments

None.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The Contact PreDeleteRecord Method

This method implements a reaction to the event of pre-deleting a record from the Contact-containing business components—for example, `Contact`, `Employee`, and others. It calls the `The OnContactPreDeleted Method`. It is important that this method is called before a contact record is finally deleted from the business component.

This method is designed to be called by the out-of-the-box Siebel runtime event handlers. It uses the `Profile` attributes to store the input and output data. These `Profile` attributes are set by the runtime event handlers when the appropriate business component's fields are changed.

Input Arguments

None.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnCampaignAssociated Method

This method generates synchronization requests that must be performed when the campaign is enabled for synchronization.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnCampaignDisassociated Method

This method generates synchronization requests that must be performed when the campaign is disabled for synchronization.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnCampaignLoaded Method

This method generates synchronization requests that must be performed when the campaign load is loaded.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnCampaignPurged Method

This method generates synchronization requests that must be performed when the campaign load is purged.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnWaveLaunched Method

This method generates synchronization requests that must be performed when the campaign load wave is launched.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.
`PrevState`	Optional: The previous wave status. It is important to know whether the wave was suspended before launching or not. The value must correspond to the language-independent code (LIC) for the `Status` field of the `Campaign Load Wave` business component or an empty strings. By default, the value is an empty string.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnWaveSuspended Method

This method generates synchronization requests that must be performed when the campaign load wave is suspended.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnWaveUpdated Method

This method generates synchronization requests that must be performed when the campaign load wave is updated.

Input Arguments

`CampaignId`	Mandatory: The ID of the Siebel campaign. This value must correspond, for example, to the `Campaign Id` field of the `Campaign List Contact` business component.
`LoadNumber`	Mandatory: The load number of a Siebel campaign. This value must correspond to the `Load Number` field of the `Campaign List Contact` business component.

`WaveNumber`	Mandatory: The wave number of a Siebel campaign load. This value must correspond to the `Wave Number` field of the `Campaign List Contact` business component.
`WaveStatus`	Mandatory: The wave status of a Siebel campaign load. This value must correspond to the language-independent code (LIC) for the `Status` field of the `Campaign Load Wave` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnContactUpdated Method

This method generates synchronization requests that must be performed when the Siebel contact or prospect is updated. The requests are generated for each campaign and wave (calling list) that contains the contact information.

Input Arguments

`ContactId`	Mandatory: The ID of the Siebel contact or prospect. This value must correspond to the `Calculated Contact Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The OnContactPreDeleted Method

This method generates synchronization requests that must be performed when the Siebel contact or prospect is deleted. The requests are generated for each campaign and wave (calling list) that contains the contact information.

Note: This method must be called before the contact or prospect is finally deleted.

Input Arguments

`ContactId`	Mandatory: The ID of the Siebel contact or prospect. This value must correspond to the `Calculated Contact Id` field of the `Campaign List Contact` business component.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The Genesys CampSynch Utils Business Service

The `Genesys CampSynch Utils` business service implements various over-all helpful utilities that can be used in the Campaign Synchronization solution.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Utils` business service:

- The BuildGenesysCampaignName Method, page 86
- The BuildGenesysCallingListName Method, page 87
- The IsCfgObjWriteAllowed Method, page 87
- The IsCfgObjDeletionAllowed Method, page 88
- The IsWFProcessRunning Method, page 88
- The RunWFProcess Method, page 89
- The StopWFProcess Method, page 89
- The CalcSiebelSideWaveSynchStatus Method, page 90

The BuildGenesysCampaignName Method

This method builds the name of the Genesys campaign, based on the Siebel name and the load number.

Input Arguments

`SiebelCampaignName`	Mandatory: The name of a Siebel campaign.
`LoadNumber`	Mandatory: The load number of a Siebel campaign.

Output Argument

`GenesysCampaignName`	Mandatory: The name of a Genesys campaign.

Exceptions

This method raises exceptions if there are errors.

The BuildGenesysCallingListName Method

This method builds the name of the Genesys Calling List, based on the Siebel campaign name, the load number, and the wave number.

Input Arguments

`SiebelCampaignName`	Mandatory: The name of a Siebel campaign.
`LoadNumber`	Mandatory: The load number of a Siebel campaign.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign.

Output Argument

`GenesysCallingList Name`	Mandatory: The name of a Genesys Calling List.

Exceptions

This method raises exceptions if there are errors.

The IsCfgObjWriteAllowed Method

This method checks if a `CampSynch Configuration` object has any valid values to be saved or not.

Input Arguments

`CasConfObjId`	Mandatory: The ID of a `CampSynch Configuration` object.
`ProfilePrimary`	Mandatory: The new name of the primary connection subsystem (the Siebel profile).
`ProfileBackup`	Mandatory: The new name of the backup connection subsystem (the Siebel profile).

Output Argument

`ResultMessage`	An empty string if record deletion is allowed, or a reason why record deletion is not allowed.

Exceptions

This method raises exceptions if there are errors.

The IsCfgObjDeletionAllowed Method

This method checks if a `CampSynch Configuration` object is allowed to be deleted or not.

Input Arguments

`CasConfObjId`	Mandatory: The ID of a `CampSynch Configuration` object.
`CampWFStatus`	Mandatory: The `Campaign Workflow` running status (either `Running` or `Shutdown`).
`CRWFStatus`	Mandatory: The `CR Workflow` running status (either `Running` or `Shutdown`).

Output Argument

`ResultMessage`	An empty string if record deletion is allowed, or a reason why record deletion is not allowed.

Exceptions

This method raises exceptions if there are errors.

The IsWFProcessRunning Method

This method checks if a specified workflow process is running for a specified `CampSynch Configuration` object or not.

Input Arguments

`CasConfObjId`	Mandatory: The ID of a `CampSynch Configuration` object.
`WFName`	Mandatory: The workflow name. The out-of-the-box solution names are `Genesys CaS CR WF` and `Genesys CaS campaign WF`.

Output Argument

`IsRunning`	`Y` or `N`.

Exceptions

This method raises exceptions if there are errors.

The RunWFProcess Method

This method asynchronously runs a specified workflow process for a specified `CampSynch Configuration` object.

Input Arguments

`CasConfObjId`	Mandatory: The ID of a `CampSynch Configuration` object.
`WFName`	Mandatory: The workflow name. The out-of-the-box solution names are `Genesys CaS CR WF` and `Genesys CaS campaign WF`.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The StopWFProcess Method

This method stops a specified workflow process for a specified `CampSynch Configuration` object or not.

Input Arguments

`CasConfObjId`	Mandatory: The ID of a `CampSynch Configuration` object.
`WFName`	Mandatory: The workflow name. The out-of-the-box solution names are `Genesys CaS CR WF` and `Genesys CaS campaign WF`.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The CalcSiebelSideWaveSynchStatus Method

This method calculates the Siebel-oriented status of a campaign wave synchronization process. This status displays the synchronization status based on the information that is present in the Siebel-side only. It is not aware of what is happening on the Genesys-side.

The resulting status values correspond to the LOV records that have the `GENESYS_SBL_SYNCH_STATUS` type.

Input Arguments

`SiebelCampaignName`	Mandatory: The name of the Siebel campaign.
`LoadNumber`	Mandatory: The load number of a Siebel campaign.
`WaveNumber`	Mandatory: The wave number of a Siebel campaign load.

Output Argument

`StatusLIC`	The status value (language-independent code) for the synchronization status.

Exceptions

This method raises exceptions if there are errors.

The Genesys CampSynch Tools Business Service

The `Genesys CampSynch Tools` business service provides methods that help to deploy the Campaign Synchronization solution.

The Method Descriptions

The entries in this section describe the following methods of the `Genesys CampSynch Tools` business service:

- The InstallRuntimeEvents Method, page 90
- The UninstallRuntimeEvents Method, page 91
- The ImportAll Method, page 91

The InstallRuntimeEvents Method

This method installs the Siebel runtime events handlers in the Siebel Server and is designed to run using the Siebel Business Service Simulator.

Input Arguments

None.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The UninstallRuntimeEvents Method

This method uninstalls the Siebel runtime events handlers in the Siebel Server and is designed to run using the Siebel Business Service Simulator.

Input Arguments

None.

Output Argument

None.

Exceptions

This method raises exceptions if there are errors.

The ImportAll Method

This method imports the required Siebel `LOV` records and the `Responsibility` records to the Siebel Server. It is designed to run using the Siebel Business Service Simulator with inputs that are stored in a special `*.xml` file. The out-of-the-box solution provides the `GplusCaS_LOV.xml` file as an input for this method.

Input Arguments

The input arguments have a complex structure, so use the provided `*.xml` file that stores these arguments.

Output Argument

Use the report strings relevant to the imported records.

Exceptions

This method raises exceptions if there are errors.

Chapter

2 Media Routing Component Customization

The G*plus* Media Routing for Siebel CRM Component (Media Routing Component) integrates the Siebel software and the Genesys software's handling of open media interactions.

This chapter describes the Media Routing Component's interfaces and customizations in the following sections:

Note: The G*plus* Media Routing for Siebel CRM and G*plus* iWD Routing for Siebel CRM components contain common parts of the code. It is not recommended to use both of these components simultaneously.

Overview

The principal parts of the Media Routing Component 8.0 are shown in Figure 1.

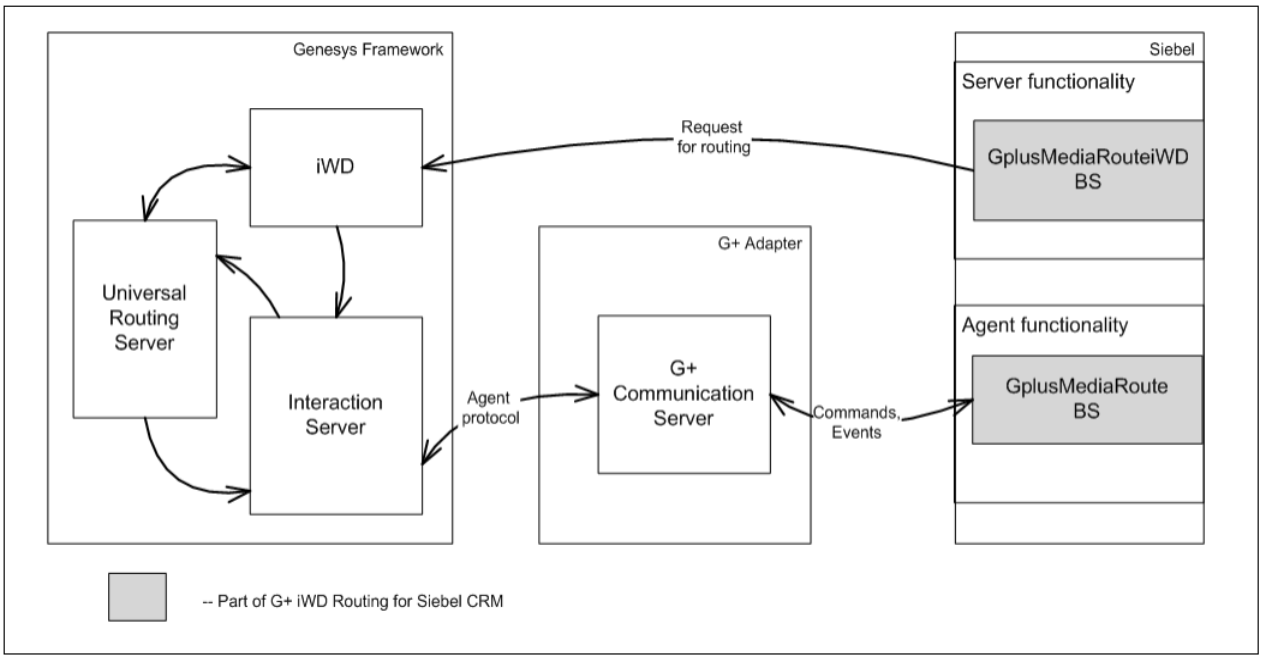

Figure 1: The Principal Parts of the Media Routing Component

In general, the Media Routing Component consists of two functions:

- The support of server functionality, which provides the ability to route interactions and to stop routing by request from the Siebel side.
- The support of agent functionality, which provides the ability to handle an interaction during an agent session.

The server functionality is provided by the G*plus* Open Media Server (a standalone Genesys application) that submits requests to the Interaction Server through the *media server* protocol and the `GplusMediaRoutingIXN` Siebel business service that enables the formation and the distribution of requests to the G*plus* Open Media Server.

The agent functionality is provided by the `GplusMediaRoute` Siebel business service, which allows the pulling of an interaction, marking it as complete, and updating the Siebel object that corresponds to the interaction.

The Media Routing Component supports a link between Siebel activities and Genesys interactions by means of a one-to-one relationship between the `InteractionId` and the `ThirdPartyId`, where `InteractionId` represents a Genesys interaction ID, and `ThirdPartyId` represents a Siebel activity record ID (or another object—for example, `ServiceRequest`). A special field in the `ThirdPartyId` that represents the Siebel object is used to store a Genesys

InteractionId—for example, the Call Id field is used for Siebel eMail) and the Genesys interaction should contain the ThirdPartyId in the attached data.

The relationship between the Siebel activities and the Genesys interactions is essential for the Pull/Stop functionality to work. However, if you do not use this functionality, you may choose not to support this association in your customization (for example—you may choose not to store a Genesys Interaction Id in a Siebel eMail activity record). Regardless of whether you use the Pull/Stop functionality or note, each Genesys interaction must have a ThirdPartyId. Please see the description of the "The Route Method" on page 96. For your individual customization, you may use any unique Siebel record field value as the ThirdPartyId.

For example—when a GplusMediaRouting-ProcessMessage workflow sends a routing request, it writes an InteractionId in the Call Id field of the Action Business Component and changes the status of the Siebel activity. If it is a successful routing request, the activity status will be Queued; otherwise it will be NotQueued.

For your customization, you may use any field instead of the Call Id field. If a Siebel Business Component does not have a spare or reusable field to be used for a Genesys Interaction Id, you must add a new field into the business component using the custom extension columns or an extension table. For more information, please refer to the *Siebel Tools Reference* from Siebel.

Using the GplusMediaRouteIXN Business Service

The GplusMediaRouteIXN business service is a business service which provides server functionality and allows submitting requests for routing an interaction and stopping its processing. The GplusMediaRouteIXN business service contains the following methods:

- The Route Method
- The StopWorkItem Method

Note: To enable a debug log, you may add a DebugLogFile input parameter and set it to a debug log file name.

The following tables describe the available methods and their arguments:

- Table 22, "The Route Method Input Parameters," on page 97
- Table 23, "The StopWorkItem Method Input Parameters," on page 102

The following symbols represent the different methods and arguments found in Table 22 and Table 23.:

`M:`	Represents mandatory arguments.
`M*:`	Represents parameters that must either be passed as an argument or have a default value set in the business service user properties.
`M**:`	Represents parameters that are required to update the Siebel record status.
`O:`	Represents the optional arguments.

The Route Method

The `route` method is used to send a routing request to the Genesys environment. The `route` method gathers all of the necessary input parameters and sends a routing request to a G*plus* Open Media Server. If you would like to add `AttachedUserData` values to a routing request, you should add the custom input arguments. All input arguments, except the predefined arguments, are attached to a routing request as a `AttachedUserData` value. The method returns the `Result`, `ErrorText`, `RouteResult`, and `RouteMessage` parameters. Also, this method can update the Siebel object record status, if required.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` (zero) indicates a failure. The `Result` value was introduced, in addition to the `RouteResult` value, since it is the standard Siebel output parameter that is used in CTI communications.

The `RouteMessage` parameter contains the Genesys `InteractionId`.

The `ErrorText` parameter contains the error description, if there is any.

Note: If the `route` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The `route` method results depend on the direct routing results only.

For a list of input parameters and their descriptions, see Table 22 on page 97.

Table 22: The Route Method Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The `RecordId` for the routing interaction. There is no default value.
PrimaryGOMSConnectionSubsystem	M	`PrimaryGOMS Connection Subsystem`	The name of the Siebel `HTTPSubSys`, created for a primary G*plus* Open Media Server. See section *"Creating a Connection Subsystem"* in Chapter 8 of the *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide*.
BackupGOMSConnectionSubsystem	O	`BackupGOMS Connection Subsystem`	The name of the Siebel `HTTPSubSys`, created for a backup G*plus* Open Media Server.
ConnectionName	M		The `Connection` name (the `Application` object of the Interaction Server) as specified in the `Connections` tab for the G*plus* Open Media Server. If you use this method as a communication profile command, you may skip the `ConnectionName` parameter, as this method reads the parameter value from communication profile parameters. If you use this method in a workflow, you *must* set the parameter value.
RoutingMediaType	M		The media type of interaction. Only the values configured in Genesys Framework are permissible. There is no default value.

Table 22: The Route Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
InteractionType	O		The `Interaction` type. The default value is `Inbound`. **Note:** Do *not* change the default value *unless* requested by Genesys Tech Support.
InteractionSubtype	O		The `Interaction` subtype. The default value is `InboundNew`. **Note:** Do *not* change the default value *unless* requested by Genesys Tech Support.
SubmitQueue	M		The name of the queue where the interaction is submitted. If you use this method as a communication profile command, you may skip the `MediaRoutingDefaultQueue` parameter, as this method reads the parameter value from the communication profile parameters. If you use this method in a workflow, you *must* set the parameter value. There is no default value.
RefreshViewAfterUpdate	O		If this parameter's value equals `Yes` and the method is called from the Communications Configuration command, the active view is refreshed after the routing request is sent to show the resulting record update. There is no default value.

Table 22: The Route Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
ReceivedAt	O		The timestamp (date and time) that is displayed when an interaction is received. The default value is a current timestamp at the moment of the method call.
BusObject	M**		The Siebel Business Object name. If this Business Object name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
BusComp	M**		The Siebel Business Component name. If this Business Component name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
StatusField	M**		The name of the Business Component field that contains status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains substatus information. There is no default value.

Table 22: The Route Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
SuccessStatus	O		The value of the record's status if the operation is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if the operation is successful. There is no default value.
FailedStatus	O		The value of the record's status if the operation has failed. There is no default value.
FailedSubStatus	O		The value of the record's substatus if the operation has failed. There is no default value.
UserField	O		The name of the Business Component field that contains a user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument provided. There is no default value.

The StopWorkItem Method

The `StopWorkItem` method is used to cancel a route request, stopping it from being processed in the Genesys environment. This method may also update the status of a Siebel record and assign it to the agent. Refer to Table 23 on page 102 for details.

The `StopWorkItem` method gathers all the necessary input parameters and sends a `StopWorkItem` request to a G*plus* Open Media Server. This method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `RouteResult` value of `0` (zero) indicates a successful operation; any value that is not `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` (zero) indicates failure. This value was introduced in addition to the `RouteResult` value since it is a standard Siebel output parameter that is used in CTI communications.

The `ErrorText` parameter contains an error description, if there are any present.

Note: If the `StopWorkItem` method is configured to update the Siebel object record, the result of the update (whether successful or not) does not affect the overall result. The `StopWorkItem` method results depend on the direct request results only.

For a list of input parameters and their descriptions, see Table 23 on page 102.

Table 23: The StopWorkItem Method Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The `RecordId` for the search interaction to be stopped. There is no default value. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided.
InteractionId	M		The `InteractionId` of the interaction to be stopped. Either the `ThirdPartyId` or the `InteractionId` parameters must be provided. If there is no value provided, the value is found by searching the Business Component using the `ThirdpartyId` value.
PrimaryGOMSConnection Subsystem	M*	`PrimaryGOMS Connection Subsystem`	The name of the Siebel `HTTPSubSys`, created for a primary G*plus* Open Media Server. See section *"Creating a Connection Subsystem"* in Chapter 8 of the *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide*.
BackupGOMSConnection Subsystem	O	`BackupGOMS Connection Subsystem`	The name of the Siebel `HTTPSubSys`, created for a backup G*plus* Open Media Server.

Table 23: The StopWorkItem Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
ConnectionName	M		The `Connection` name (the `Application` object of the Interaction Server) as specified in the `Connections` tab for the G*plus* Open Media Server. If you use this method as a communication profile command, you may skip the `ConnectionName` parameter, as this method reads the parameter value from the communication profile parameters. If you use this method in a workflow, you *must* set the parameter value.
Reason	O		The `Reason` value to be passed to the Interaction Server's `StopWorkItem` method. There is no default value.
Description	O		The `Description` value to be passed to the Interaction Server's `StopWorkItem` method. There is no default value.
RecIdField	M**		The name of a Business Component field that contains the Siebel `RecordID`. The default value is `Id`.

Table 23: The StopWorkItem Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
BusObject	M		The Siebel Business Object name. If this Business Object name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
BusComp	M	`Action`	The Siebel Business Component name. If this Business Component name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
InteractionField	M		The name of a Business Component field that contains the Genesys `InteractionId`. There is no default value.
StatusField	M**		The name of a Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of a Business Component field that contains substatus information. There is no default value.
SuccessStatus	O		The value of a record's status in the case of a successful operation. There is no default value.

Table 23: The StopWorkItem Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
SuccessSubStatus	O		The value of a record's substatus in the case of a successful operation. There is no default value.
FailedStatus	O		The value of a record's status in the case of a failed operation. There is no default value.
FailedSubStatus	O		The value of a record's substatus in the case of a failed operation. There is no default value.
UserField	O		The name of a Business Component field that contains a user name. There is no default value.
UserName	O		The user name that is set for a Siebel record if a `UserField` argument is provided. There is no default value.

Using the GplusMediaRoute Business Service

The GplusMediaRoute business service provides agent functionality, allows for the pulling and marking of an interaction as completed, and updates the Siebel record.

The GplusMediaRoute business service contains the following methods:

- The GetTopWorkItem Method
- The GetTopActiveItemInfo Method
- The MarkWorkItemDone Method
- The PullInteraction Method
- The UpdateActivity Method

The following tables describe the available methods and their arguments:

- Table 24, “The MarkDone Methods Input Parameters,” on page 108
- Table 25, “The PullInteraction Method Input Parameters,” on page 111
- Table 26, “The UpdateActivity Method Input Parameters,” on page 113

The following symbols, found in the tables listed above, represent the different methods and arguments:

M:	Represents mandatory arguments.
M*:	Represents parameters that must either be passed as an argument or have a default value set in the business service user properties.
M**:	Represents parameters that are required to update the Siebel record status.
O:	Represents the optional arguments.

Note: To enable a debug log, you may add a DebugLogFile input parameter and set it to a debug log file name.

The GetTopWorkItem Method

The `GetTopWorkItem` method is used to get the `ThirdPartyId` parameter of a top active work item.

The single input optional parameter is `MediaType`. If the `MediaType` parameter is provided, then the `ThirdPartyId` is returned only if top active work item's media type is equal to its value.

The output parameter is `ThirdPartyId`. For the meaning of these parameters see the "Overview" on page 94.

The GetTopActiveItemInfo Method

The `GetTopActiveItemInfo` method is used to get information about a top active work item.

It does not have any input parameters.

The output parameters are `DriverWorkTrackID` (that corresponds to `Interaction Id`), `MediaType`, and `ThirdPartyId`.

The MarkWorkItemDone Method

The `MarkWorkItemDone` method is used to mark a Siebel work item as done.

If the `QueueName` input parameter is set then it is used; otherwise, the `QueueParameterName` parameter is used and the provided parameter is read from the CTI Communications configuration.

If the `QueueParameterName` input parameter name is not set, then the default value `MediaRoutingDoneQueue` is used. If there is no configuration parameter found, then a special predefined queue `__STOP__` is used.

The `MarkWorkItemDone` method sets a `MarkDoneQueue` output parameter and invokes a `MarkDoneMR` command from the CTI Communications configuration.

Note: The `MarkDoneMR` command uses the `{@SelectedWorkItem:DriverWorkTrackID}` Siebel macros, so this command is applied to a selected work item.

The output parameter values are `Result` and `ErrorText`.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates a failed operation.

For a list of the input parameters and their descriptions, see Table 24 on page 108.

Table 24: The MarkDone Methods Input Parameters

Argument Name	Symbol	Default User Property Name	Description
QueueName	O		The name of the queue where the interaction should be placed. The default value is `__STOP__`. This value is used if: • neither `QueueName`, nor `QueueParameterName` is provided • there is no CTI parameter found • if it is empty.
QueueParameterName	O		The CTI Communications parameter name that contains the queue name where an interaction should be placed. This value is used if there is no `QueueName` provided. The default value is `MediaRoutingDoneQueue`.
ThirdPartyId	M**		The `RecordId` for a search item of an interaction that is to be marked as done. There is no default value. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided.
InteractionId	M**		The `InteractionId` parameters *must* be provided. If there is no value provided, the value is found by searching the business component by using the `ThirdpartyId` value.
BusObject	M**		A Siebel Business Object name. If this Business Object name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.

Table 24: The MarkDone Methods Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
BusComp	M**		A Siebel Business Component name. If this Business Component name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
RecIdField	M**		The name of a Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of a Business Component field that contains the Genesys `InteractionId`. There is no default value.
StatusField	M**		The name of a Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of a Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of a record's status in a case of a successful operation. There is no default value.
SuccessSubStatus	O		The value of a record's substatus in the case of a successful operation. There is no default value.
FailedStatus	O		The value of a record's status in the case of a failed operation. There is no default value.
FailedSubStatus	O		The value of a record's substatus in the case of a failed operation. There is no default value.

Table 24: The MarkDone Methods Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
UserField	O		The name of a Business Component field that contains a user name. There is no default value.
UserName	O		A user name that is set for a Siebel record if the `UserField` argument is provided. There is no default value

The PullInteraction Method

The `PullInteraction` method is used to pull an interaction from a queue while it is in the queue waiting for processing.

Note: The interaction can not be pulled, if the interaction is being processed by another agent or by Genesys Universal Routing.

If the `InteractionId` input parameter is set, it is used; otherwise the `ThirdPartyId` is used to find a proper `InteractionId`. The `PullInteraction` method opens a `BusComp` from the `BusObject`, locates a record where the `RecIdField` field equals `ThirdPartyId`, and uses an `InteractionField` field as the `InteractionId`. This method sets the `InteractionId` output parameter and invokes the `OpenMediaPullInteractionById` command from the Communications configuration.

The output parameter values are `Result` and `ErrorText`.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates a failed operation.

For a list of input parameters and their descriptions, see Table 25.

Table 25: The PullInteraction Method Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The `RecordID` for the search interaction to be pulled. There is no default value. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided.
InteractionId	M		The `InteractionId` to be used to pull the interaction. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided. If there is no value provided, the value is found by searching the business component by using the `ThirdpartyId` value.
ThirdPartyId	M		The ID to be used as the Siebel `RecordID`. See the "RecIdField" on page 112.
BusObject	M (if no `InteractionID` is provided)		The Siebel Business Object name. If this Business Object name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
BusComp	M (if no `InteractionID` is provided)		The Siebel Business Component name. If this Business Component name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.

Table 25: The PullInteraction Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
InteractionField	M (if no `InteractionID` is provided)		The name of a Business Component field that contains the Genesys `InteractionId`. There is no default value.
RecIdField	M (if no `InteractionID` is provided)		The name of a Business Component field that contains the Siebel `RecordID`. The default value is `Id`.

The UpdateActivity Method

The `UpdateActivity` method is used to update Siebel records. Using this method, it is possible to update the following values:

- `Interaction Id`: the value that is that stored in a field reference by the `InteractionField` parameter.
- `Status`: the value that is stored in a field referenced by the `StatusField` parameter.
- `Substatus`: the value that is that stored in a field referenced by the `SubStatusField` parameter.
- `User name`: the value that is stored in a field referenced by the `UserField` parameter.

The output parameter values are `Result` and `ErrorText`.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates a failed operation.

For a list of input parameters and their descriptions, see Table 26.

Table 26: The UpdateActivity Method Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The `RecordID` for the search interaction to be pulled. There is no default value. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided.
InteractionId	M		The `InteractionID` to be used to pull the interaction. Either the `ThirdPartyId` or the `InteractionId` parameters *must* be provided. If there is no value provided, the value is found by searching the business component by using the `ThirdpartyId` value.
BusObject	M		The Siebel Business Object name. If this Business Object name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
BusComp	M		The Siebel Business Component name. If this Business Component name is provided, an attempt to update the record status (for example—`Activity`) is made. There is no default value.
RecIdField	M		The name of a Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M		The name of a Business Component field that contains the Genesys `InteractionId`. There is no default value.

Table 26: The UpdateActivity Method Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
StatusField	M		The name of a Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of a Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of a record's status in a case of a successful operation. There is no default value.
SuccessSubStatus	O		The value of a record's substatus in the case of a successful operation. There is no default value.
FailedStatus	O		The value of a record's status in the case of a failed operation. There is no default value.
FailedSubStatus	O		The value of a record's substatus in the case of a failed operation. There is no default value.
UserField	O		The name of a Business Component field that contains a user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument is provided. There is no default value.

The Applet Customization

The Media Routing Component customizes the `Comm Outbound Item Form` applet by adding some script code in the function `WebApplet_InvokeMethod (MethodName)` server scripts. If `MethodName` equals `EmailSend` or `EmailCancel`, the script calls a `MarkDoneMR` command from the Communications configuration to mark an interaction as done and removes it from the list of active work items.

If you use the Media Routing Component for routing an interaction that is different from Siebel eMail, you should add similar customization to the appropriate applet. Or, you may add a button on the toolbar or a menu item in the Communications menu to mark an active interaction as done.

Invoking a MarkDoneMR Command

You may invoke a `MarkDoneMR` command from the Communications configuration as follows:

Example 1

```
var outQueue = "Mark_Done_Queue_Name";
var ctiSvc = TheApplication().GetService("Communications Client");
var inp = TheApplication().NewPropertySet();
var outp = TheApplication().NewPropertySet();
inp.SetProperty("MarkDoneQueue", outQueue);
ctiSvc.InvokeMethod("MarkDoneMR ", inp,outp);
```

Or, you may invoke a `MarkWorkItemDone` method of the `GplusMediaRoute` business service:

Example 2

```
var outQueue = "Mark_Done_Queue_Name";
var ctiSvc = TheApplication().GetService("GplusMediaRoute ");
var inp = TheApplication().NewPropertySet();
var outp = TheApplication().NewPropertySet();
inp.SetProperty("QueueName", outQueue);
ctiSvc.InvokeMethod("MarkWorkItemDone ", inp,outp);
```

Routing Siebel Work Items

The Media Routing Component for Siebel can be used for routing any type of Siebel work items, both in real-time and in background (non real-time) modes. The Media Routing Component provides the basic functionality for Siebel work item routing, but customization is required to provide a graphical user interface (GUI). See the *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide* for general information about configuring the Media Routing Component.

The `Group` buttons on the Communications toolbar work for all interaction types—for example, `Logon/Logout`, `Accept`, and `Ready/NotReady`. However, you should create `Ready` and `NotReady` commands for custom media types and then add them into the proper command groups—for example, the `ReadyGroup` command group and the `NotReadyGroup` command group.

Creating Commands for Custom Media Types

Example 3 is a command sample.

Note: A media type is set with the prefix `@` in both the `DeviceCommand` and the `FilterSpec` parameters.

Example 3

```
[Command:ReadyForSiebelSRGroup]
FilterSpec = "[$GetCommandStatus(@ServiceRequest@OpenMediaReady)] =
              'Enabled'"
Hidden = "FALSE"
DeviceCommand = "@ServiceRequest@OpenMediaReady"
Description = "Set ready for SiebelSR media type"
Profile = " Gplus Universal Profile "

[Command:NotReadyForSiebelSR]
FilterSpec = "[$GetCommandStatus(@ServiceRequest@OpenMediaNotReady)] =
              'Enabled'"
Hidden = "FALSE"
DeviceCommand = "@ServiceRequest@OpenMediaNotReady"
Description = "Set SiebelSR media type"
Profile = " Gplus Universal Profile "
```

Using the Route Method to Send a Routing Request

To send a routing request, you must use the `route` method of the `GplusMediaRouteIXN` business service. Example 4 is a sample route command to route a service request. To use it, set the `SubmitQueue` parameter. If you send a routing request in a workflow, you must also set the `ConnectionName` parameter value. See "The Route Method" on page 96 for more information.

Example 4

```
[Command:SendRouteSR]
   Description   = "Route Service Request"
   Title         = "Route Service Request"
   ServiceMethod = "GplusMediaRouteIXN.route"
   Comments      = "Send route request to route Siebel ServiceRequest"
   Hidden        = "False"
   AllViews      = "False"
   View          = "Personal Service Request List View"
   View          = "All Service Request List View"
   View          = "Service Request Detail View"
   Profile       = "Gplus Universal Profile"
   CmdData       = "SendRouteCmdSR"
   MenuPosition  = "30.1"

[CmdData:SendRouteCmdSR]
      RequiredField.SR Number                    = "?*"
      ServiceParam.Subject                       = "{Abstract}"
      ServiceParam.PrimaryGOMSConnectionSubsystem = "
              GplusOpenMediaServerPrimary "
      ServiceParam.BackupGOMSConnectionSubsystem  = "
              GplusOpenMediaServerBackup "
      ServiceParam.BusComp                       = "Service Request"
      ServiceParam.BusObject                     = "Service Request"
      ServiceParam.InteractionField              = "Integration Id"
      ServiceParam.RecIdField                    = "SR Number"
      ServiceParam.RoutingMediaType              = "ServiceRequest"
      ServiceParam.SR_Type                       = "{SR Type}"
      ServiceParam.SubmitQueue                   = "Siebel Inbound"
      ServiceParam.ThirdPartyId                  = "{SR Number}"
      ServiceParam.StatusField            = "Status"
      ServiceParam.SubStatusField         = "Sub-Status"
      ServiceParam.RefreshViewAfterUpdate  = "Yes"
      Comments                            = ""
```

The ServiceParam.ThirdPartyId parameter should be set to the ID field for a Siebel work item record; the parameter should have the same fields as a proper event handler. In Example 4, the SR Number field value is used as the ThirdPartyId parameter.

Note: The InteractionField parameter is set in Example 4 so that it is possible to use the Pull/Stop functionality. As the SuccessStatus parameter is absent, this method does not update the record status. If you want to do a record status update, you may add the SuccessStatus and FailedStatus parameters. For more information, see "The Route Method" on page 96.

In Example 4, the ServiceParam.Subject is an optional parameter, and it is added as an AttachedUserData value. However, this value is used in a work item description text message. See the itx_scdrv.xml file in the G*plus* Communication Server folder. For more information, see "Using the GplusMediaRouteIXN Business Service" on page 95.

Creating an Event Handler

Procedure: Creating an event handler

Purpose: To create an event handler for each custom media type to open a Siebel view for the routed work item.

Start of procedure

1. To accept a routed work item, the agent must click the Accept group button or the Accept Multimedia Interaction sub-button.
2. Create an event handler for the OpenMediaAccepted event for each custom media type to open a Siebel view for the routed work item.
3. Set the QuerySpec parameter to the value:

   FieldName='{ThirdPartyId}'

   where FieldName is the ID field for a Siebel work item.
4. Set the SingleView parameter to a proper Siebel view name. See Example 5 and Example 6:

Example 5

```
[EventHandler:OpenMediaSelectedSR]
   Filter.MediaType  = "ServiceRequest"
   Profile           = "Gplus Universal Profile"
   Comments          = "EventHandler samples  for Siebel work items
               routing"
```

```
   Order             = "50"
   Response          = "OpenSiebelViewSR"
   DeviceEvent       = "OpenMediaSelected"

[EventResponse:OpenSiebelViewSR]
      QueryBusComp   = "Service Request"
      QueryBusObj    = "Service Request"
      QuerySpec      = "SR Number='{ThirdPartyId}'"
      SingleView     = "Service Request Detail View"
      Comments       = "EventResponse samples for Siebel work items
             routing"
```

If you want to update a record status and/or assign it to the agent when an agent accepts the interaction, you should create an event handler as shown in Example 6:

Example 6

```
[EventHandler:OpenMediaAcceptedSR]
   Filter.ThirdPartyId = "?*"
   Filter.MediaType    = "ServiceRequest"
   DeviceEvent         = "OpenMediaAccepted"
   Profile             = "Gplus Universal Profile"
   Response            = "EventResponseAcceptSR"
   Order               = "50"

[EventResponse:EventResponseAcceptSR]
      QueryBusComp = "Service Request"
      QueryBusObj  = "Service Request"
      Log          = "EventLogAcceptSR"

[EventLog:EventLogAcceptSR]
         BusComp                  = "Service Request"
         BusObj                   = "Service Request"
         LogField.Owner           = "{@UserName}"
         LogField.Status          = "CHANGE_ME"
         LogField.Sub-Status      = "CHANGE_ME"
         QuerySpec                = "SR Number='{ThirdPartyId}'"
```

End of procedure

Next Steps

- There are no further steps.

You have to provide a command to mark a work item as done. For this you may configure the provided `MarkWorkItemDone` command to support your

custom types, such as adding the media types into the `FilterMediaType` parameter value, which contains a list of media types separated by commas.

As an alternative, you may customize a button on a work item view or add a custom button on the communication toolbar to invoke a `MarkDone` command. Refer to the section "The Applet Customization" on page 115.

The provided out-of-the-box sample shows how to route Siebel Service Requests in real-time mode and how to route Service Orders in background mode. It uses the `ServiceRequest` media type for Siebel Service Requests and the `ServiceOrder` media type for Service Orders.

Procedure: Using the provided sample to route Siebel service request

Purpose: To use the provided sample to route Siebel Service Requests in real-time mode and how to route Service Orders in background mode.

Start of procedure

1. Remove the comment marks in the `GenComm_universal.def` (appropriate places have the following comment: "`Use this command in couple with Gplus iWD Routing for Siebel CRM`") file before importing a configuration and performing the above-mentioned actions.
2. Create a custom media type in the Genesys environment and then add in the `Channel String` parameter of the `Gplus Universal Profile` profile driver.
3. Set the proper values for the command and events used.

End of procedure

Next Steps

- There are no further steps.

For more information about the Media Routing Component deployment and configuration, please refer to Chapter 8 Deploying the Media Routing Component in the *G*plus *Adapter 8.0 for Siebel CRM Deployment Guide*.

Chapter

3 iWD Routing Component Customization

The G*plus* iWD Routing for Siebel CRM Component (iWD Routing Component) integrates the Siebel and Genesys software's handling of interactions.

Note: The *Gplus* iWD Routing for Siebel CRM and *Gplus* Media Routing for Siebel CRM components contain common parts of code. It is not recommended to use both these components simultaneously.

This chapter describes the iWD Routing Component's interfaces and customizations in the following sections:

Note: The G*plus* Media Routing for Siebel CRM and G*plus* iWD Routing for Siebel CRM components contain common parts of the code. It is not recommended to use both of these components simultaneously.

Overview

The principal parts of the iWD Routing Component 8.0 are shown in Figure 2

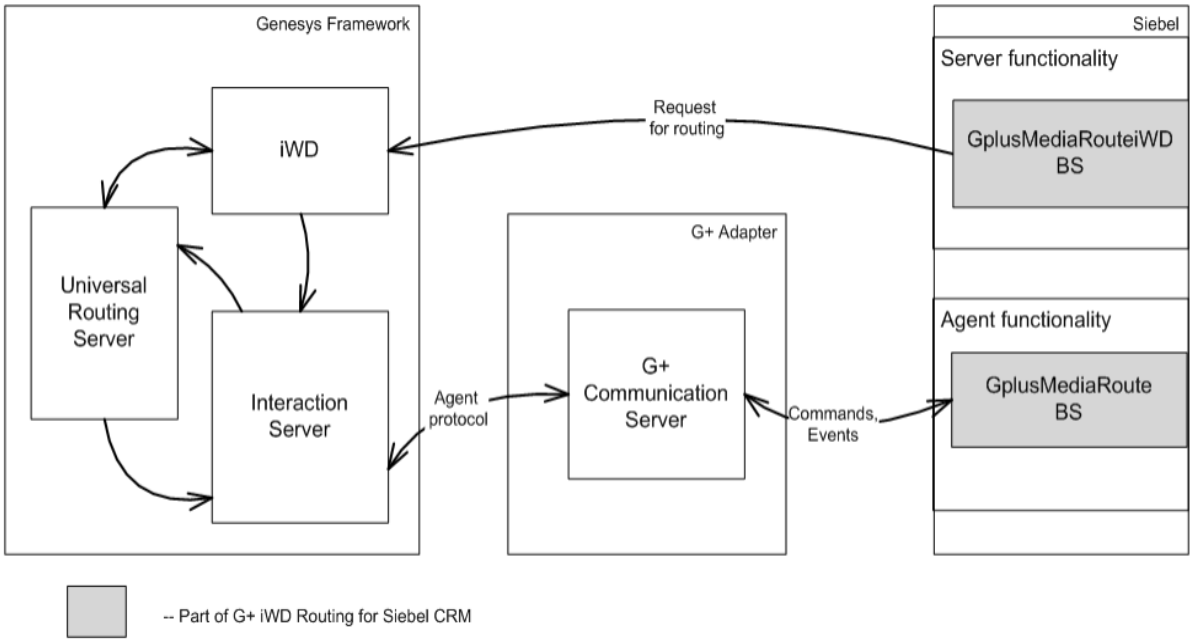

Figure 2: Principal Parts of the iWD Routing Component

In general, the iWD Routing Component consists of two functions:

- Support of server functionality. This functionality provides the ability to submit requests to the Genesys iWD solution to control the interaction routing process from the Siebel side.
- Support of agent functionality. This functionality provides the ability to handle an interaction during an agent session.

Refer to the *Gplus Adapter 8.0 for Siebel CRM Deployment Guide* for details.

The server functionality is provided by the `GplusMediaRoutingiWD` Siebel business service, which allows the formation and the distribution of requests to Genesys iWD. Additionally, there are several automatically generated objects used in tandem with the `GplusMediaRoutingiWD` business service. They represent the outbound Web service that communicates with iWD Web Service Capture Point, and they do not require any customization. Refer to the *Gplus Adapter 8.0 for Siebel CRM Deployment Guide* for details.

The agent functionality is provided by the `GplusMediaRuote` Siebel business service, which allows pulling an interaction, marking it as done, and updating the Siebel object that corresponds to the interaction.

The iWD Routing Component supports a link between Siebel activities and Genesys interactions by means of a one-to-one relationship between the `InteractionId` and the `ThirdPartyId`, where `InteractionId` represents a

Genesys InteractionID, and ThirdPartyId represents a Siebel activity RecordID (or another object, for example—Service Request). A special field in the ThirdPartyId that represents the Siebel object is used to store a Genesys InteractionId (for example—the Call Id field is used for Siebel eMail) and the Genesys interaction should contain the ThirdPartyId in the attached data.

Note: To keep semblance with Media Routing Component, the ThirdPartyId name is used as a parameter name, and it is mapped to the Capture Id iWD parameter.

The relationship between the Siebel activities and the Genesys interactions is essential for the Pull/Stop functionality to work. However, if you do not use this functionality, you may choose not to support this association in your customization (for example—you may choose not to store a Genesys InteractionId in a Siebel eMail activity record). Regardless of whether you use the Pull/Stop functionality or not, each Genesys interaction must have a ThirdPartyId. For your individual customization, you may use any unique Siebel record field value as the ThirdPartyId.

For example—when a GplusMediaRouting-ProcessMessage workflow sends a routing request, it writes an InteractionId in the Call Id field of the Action Business Component and changes the status of the Siebel activity. If it is a successful routing request, the activity status will be Queued; otherwise it will be NotQueued.

For your customization, you may use any field instead of the Call Id field. If a Siebel Business Component does not have a spare or reusable field to be used for a Genesys Interaction Id, you must add a new field into the Business Component using the custom extension columns or an extension table. For more information, see the *Siebel Tools Reference* from Siebel.

Using the GplusMediaRouteiWD Business Service

The GplusMediaRouteiWD business service provides server functionality and enables requests to be submited that are supported by Genesys iWD to control the interaction routing process.

The GplusMediaRouteiWD business service contains the following methods:

- The cancelTaskByCaptureId Method
- The completeTaskByCaptureId Method
- The createTask Method
- The getTaskByCaptureId Method
- The holdTaskByCaptureId Method
- The ping Method

- The restartTaskByCaptureId Method
- The resumeTaskByCaptureId Method
- The updateTaskByCaptureId Method

For a detailed description of these iWD methods, see the *intelligent Workload Distribution 8.0. Deployment Guide* for details.

The following tables describe the available methods and their arguments:

- Table 27, "The cancelTaskByCaptureId Input Parameters," on page 125
- Table 28, "The completeTaskByCaptureId Input Parameters," on page 128
- Table 29, "The createTask Input Parameters," on page 131
- Table 30, "The getTaskByCaptureId Input Parameters," on page 135
- Table 31, "The holdTaskByCaptureId Input Parameters," on page 136
- Table 32, "The ping Input Parameters," on page 139
- Table 33, "The restartTaskByCaptureId Input Parameters," on page 140
- Table 34, "The resumeTaskByCaptureId Input Parameters," on page 143
- Table 35, "The updateTaskByCaptureId Input Parameters," on page 147

The following symbols, found in the tables listed above, represent the different methods and arguments:

M:	Represents mandatory arguments.
M*:	Represents parameters that must either be passed as an argument or have a default value set in the business service user properties.
M**:	Represents parameters that are required to update the Siebel record status.
O:	Represents the optional arguments.

Note: To enable a debug log, you can add a DebugLogFile input parameter and set it to a debug log file name.

The cancelTaskByCaptureId Method

The cancelTaskByCaptureId method is used to send a cancelTaskByCaptureId request to the Genesys iWD. This method cancels an interaction from routing. The cancelTaskByCaptureId method gathers all the necessary input parameters and sends a cancelTaskByCaptureId request to iWD. Also, this method can update the Siebel object record status, if required.

This method returns the RouteResult, Result, and ErrorText parameters.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI Communications.

The `ErrorText` parameter contains the error description, if there is any.

Note: If the `cancelTaskByCaptureId` method is configured to update the Siebel object record, the update results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 27.

Table 27: The cancelTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The `RecordId` for the routing interaction. Used as the `CaptureId` parameter value for iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.

Table 27: The cancelTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is used only for auditing purposes and is set to `SYSTEM` if the value is not provided.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.

Table 27: The cancelTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of a record's substatus if a call made to iWD is unsuccessful. There is no default value.
UserField	O		The name of a business component field that contains the user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument is provided. There is no default value.

The completeTaskByCaptureId Method

The `completeTaskByCaptureId` method is used to send a `completeTaskByCaptureId` request to the Genesys iWD. This method completes an interaction routing process.

The `completeTaskByCaptureId` method gathers all of the necessary input parameters and sends a `completeTaskByCaptureId` request to iWD. Also, this method can update the Siebel object record status, if required.

This method returns the `RouteResult`, `Result`, and `ErrorText` parameters.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI Communications.

The `ErrorText` parameter contains the error description, if there is any.

Note: If the `completeTaskByCaptureId` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 28.

Table 28: The completeTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is used only for auditing purposes and is set to `SYSTEM`, if the value is not provided.

Table 28: The completeTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
completedDateTime	O		iWD: The date and time when the task was completed. Format is `YYYY-MM-DDThh:mm:ss`.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example— `Activity`). There is no default value.
BusComp	M**		The Siebel business component name. If provided*, an attempt is made to update the record status (for example— `Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.

Table 28: The completeTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of a record's substatus if a call made to iWD is unsuccessful. There is no default value.
UserField	O		The name of a Business Component field that contains the user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument is provided. There is no default value.

The createTask Method

The `createTask` method is used to send a `createTask` request to the Genesys iWD. This method submits an interaction into a routing process.

The `createTask` method gathers all the necessary input parameters and sends a `createTask` request to iWD. If you would like to add some `AttachedUserData` values to a request, you should add custom input arguments. All input arguments, except the predefined arguments, are attached to a `createTask` request as `AttachedUserData`. Also, this method can update the Siebel object record status, if required.

This method returns the `Result`, `ErrorText`, `RouteResult`, and `RouteMessage` parameters.

A `Result` value of 1 indicates a successful operation; a `Result` value of 0 indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI Communications.

The `ErrorText` parameter contains the error description, if there is any.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

The `RouteMessage` parameter contains the Genesys `InteractionId`.

Note: If the `createTask` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 29.

Table 29: The createTask Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.

Table 29: The createTask Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
ActivationDateTime	O		iWD: The date and time when the task becomes active. Before the task is activated, it remains queued and is not reprioritized and distributed. If this parameter is not provided, the task becomes active instantly. The format is `YYYY-MM-DDThh:mm:ss`.
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is only used for auditing purposes and is set to `SYSTEM`, if the value is not provided.
BusinessValue	M*	`defaultBusinessValue`	iWD: The business value of the task.
Category	O		iWD: The task's category. For example—`Followup`.
Channel	O	`defaultChannel`	iWD: The task's media channel. For example—Fax, Email, or Web form.
DueDateTime	O		iWD: The date and time by which the task should be completed according to SLA. The format is `YYYY-MM-DDThh:mm:ss`.
ExpirationDateTime	O		iWD: The time when the task expires and is archived. Only the tasks that have been canceled, completed, or rejected are archived. The format is `YYYY-MM-DDThh:mm:ss`.

Table 29: The createTask Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
Hold	M*	`defaultHold`	iWD: Decides whether to initially hold the task. If true, the task is created with the initial status set to `NewHeld` and is not processed any further until a subsequent call to resume the `TaskByCaptureId`.
Priority	M*	`defaultPriority`	iWD: The task priority, which is an integer number used to order tasks submitted to the distribution system.
ProcessId	O	`defaultProcessId`	iWD: The ID of the process to which the task should be assigned.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example— `Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example— `Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the business component field that contains the Genesys `InteractionID`. There is no default value.

Table 29: The createTask Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of a record's substatus if a call made to iWD is unsuccessful. There is no default value.
UserField	O		The name of a Business Component field that contains the user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument is provided. There is no default value.

The getTaskByCaptureId Method

The `getTaskByCaptureId` method is used to send a `getTaskByCaptureId` request to Genesys iWD. This method is used to request current information about the interaction.

The `getTaskByCaptureId` method gathers all the necessary input parameters and sends a `getTaskByCaptureId` request to iWD.

This method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

If the request is successful, the requested information is set as a child property.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI Communications.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

The `ErrorText` parameter contains the error description, if there is any.

For a list of input parameters and their descriptions, see Table 30.

Table 30: The getTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.

The holdTaskByCaptureId Method

The `holdTaskByCaptureId` method is used to send a `holdTaskByCaptureId` request to the Genesys iWD. This method puts an interaction on hold.

The `holdTaskByCaptureId` method gathers all the necessary input parameters and sends a `holdTaskByCaptureId` request to iWD. Also, this method can update the Siebel object record status, if required.

This method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI Communications.

The `ErrorText` parameter contains the error description, if there is any.

Note: If the `holdTaskByCaptureId` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 31.

Table 31: The holdTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.

Table 31: The holdTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is used only for auditing purposes and is set to `SYSTEM`, if the value is not provided.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.

Table 31: The holdTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of a record's substatus if a call made to iWD is unsuccessful. There is no default value.
UserField	O		The name of a Business Component field that contains the user name. There is no default value.
UserName	O		The user name that is set for the Siebel record if the `UserField` argument is provided. There is no default value.

The ping Method

The `ping` method is used to send a ping request to the Genesys iWD. This method checks if the iWD Web Service Capture Point is alive or not. The `ping` method gathers all of the necessary input parameters and sends a ping request to iWD.

The method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI communications.

The `ErrorText` parameter contains the error description, if there is any.

For a list of input parameters and their descriptions, see Table 32.

Table 32: The ping Input Parameters

Argument Name	Symbol	Default User Property Name	Description
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.

The restartTaskByCaptureId Method

The `restartTaskByCaptureId` method is used to send a `restartTaskByCaptureId` request to Genesys iWD. This method restarts the routing process for an interaction.

The `restartTaskByCaptureId` method gathers all the necessary input parameters and sends a `restartTaskByCaptureId` request to iWD. Also, this method can update the Siebel object record status, if required.

This method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI communications.

The `ErrorText` parameter contains the error description, if there is any.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

Note: If the `restartTaskByCaptureId` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 33.

Table 33: The restartTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a primary web service.

Table 33: The restartTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is only used for auditing purposes and is set to `SYSTEM`, if the value is not provided.
Hold	M*	`defaultHold`	iWD: Decides whether to initially hold the task. If the value is `true`, the task is created with the initial status set to `NewHeld` and is not processed any further until a subsequent call to resume the `TaskByCaptureId`.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel record ID. The default value is `Id`.

Table 33: The restartTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of the record's substatus if a call made to iWD is unsuccessful. There is no default value.
UserField	O		The name of the Business Component field that contains the user name. There is no default value.

Table 33: The restartTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
UserName	O		The user name that is set for the Siebel record, if the `UserField` argument is provided. There is no default value.

The resumeTaskByCaptureId Method

The `resumeTaskByCaptureId` method is used to send a `resumeTaskByCaptureId` request to Genesys iWD. This method resumes an interaction routing after it was held.

The `resumeTaskByCaptureId` method gathers all of the necessary input parameters and sends a `resumeTaskByCaptureId` request to iWD. Also, this method can update the Siebel object record status, if required.

The method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI communications.

The `ErrorText` parameter contains the error description, if there is any.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

Note: If the `resumeTaskByCaptureId` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 34.

Table 34: The resumeTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.

Table 34: The resumeTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is only used for auditing purposes and is set to `SYSTEM`, if the value is not provided.
Reason	O	`defaultReason`	iWD
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value

Table 34: The resumeTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.
SuccessStatus	O		The value of the record's status if a call made to iWD is successful. There is no default value.
SuccessSubStatus	O		The value of the record's substatus if a call made to iWD is successful. There is no default value.
FailedStatus	O		The value of the record's status if a call made to iWD is unsuccessful. There is no default value.
FailedSubStatus	O		The value of a record's substatus if a call made to iWD is unsuccessful. There is no default value.

Table 34: The resumeTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
UserField	O		The name of a Business Component field that contains the user name. There is no default value.
UserName	O		The user name that is set for the Siebel record, if the `UserField` argument is provided. There is no default value.

The updateTaskByCaptureId Method

The `updateTaskByCaptureId` method is used to send an `updateTaskByCaptureId` request to the Genesys iWD. This method updates the interaction information.

The `updateTaskByCaptureId` method gathers all of the necessary input parameters and sends an `updateTaskByCaptureId` request to iWD. If you would like to add some `AttachedUserData` values to a request, you should add custom input arguments. All input arguments, except the predefined arguments, are attached to an `updateTaskByCaptureId` request as an `AttachedUserData`. Also, this method can update the Siebel object record status, if required.

This method returns the `Result`, `ErrorText`, and `RouteResult` parameters.

A `Result` value of `1` indicates a successful operation; a `Result` value of `0` indicates an unsuccessful operation. The `Result` value was introduced in addition to the `RouteResult`, since it is the standard Siebel output parameter that is used in CTI communications.

The `ErrorText` parameter contains the error description, if there is any.

A `RouteResult` value of `0` (zero) indicates a successful operation; any `RouteResult` value other than `0` (zero) represents an error code.

Note: If the `updateTaskByCaptureId` method is configured to update the Siebel object record, the updated results (whether successful or not) do not affect the overall results. The method results depend on direct iWD call results only.

For a list of input parameters and their descriptions, see Table 35.

Table 35: The updateTaskByCaptureId Input Parameters

Argument Name	Symbol	Default User Property Name	Description
ThirdPartyId	M		The ID of a Siebel work item. Used as the `CaptureId` parameter value for the iWD.
PrimaryOutboundDispatcherBS	O	`PrimaryOutbound DispatcherBS`	The name of the business service based on `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and intended to control the Outbound Web Service, which is used as a primary web service.
BackupOutboundDispatcherBS	O	`BackupOutbound DispatcherBS`	The name of the business service based on the `CSSWSOutboundDispatcher` class that is generated during the Web Service creation (`<mediaBusService>`) and is intended to control the Outbound Web Service, which is used as a backup.
ActivationDateTime	O		iWD: The date and time when the task becomes active. Before the task is activated, it remains queued and is not reprioritized and distributed. If this parameter is not provided, the task becomes active instantly. The format is `YYYY-MM-DDThh:mm:ss.`
Actor	O	`defaultActor`	iWD: The user or system that created the task. This argument is used only for auditing purposes and is set to `SYSTEM`, if the value is not provided.
BusinessValue	M*	`defaultBusiness Value`	iWD: The business value of the task.

Table 35: The updateTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
Category	O		iWD: The task's category. For example—`Followup`.
Channel	O	`defaultChannel`	iWD: The task's media channel. For example—Fax, Email, or Webform.
DueDateTime	O		iWD: The date and time by which the task should be completed according to SLA. The format is `YYYY-MM-DDThh:mm:ss`.
ExpirationDateTime	O		iWD: The time when the task expires and is archived. Only the tasks that have been canceled, completed or rejected are archived. The format is `YYYY-MM-DDThh:mm:ss`.
Hold	M*	`defaultHold`	iWD: Used to decide whether to initially hold the task, or not to hold the task. If the argument is `true`, the task is created with the initial status set to `NewHeld` and is not processed any further until a subsequent call to resume the `TaskByCaptureId`.
Priority	M*	`defaultPriority`	iWD: The task priority, which is an integer number used to order tasks submitted to the distribution system.
Processid	O	`defaultProcessId`	iWD: The ID of the process to which the task should be assigned.
Reason	O	`defaultReason`	iWD

Table 35: The updateTaskByCaptureId Input Parameters (Continued)

Argument Name	Symbol	Default User Property Name	Description
BusObject	M**		The Siebel Business Object name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value.
BusComp	M**		The Siebel Business Component name. If provided*, an attempt is made to update the record status (for example—`Activity`). There is no default value
RecIdField	M**		The name of the Business Component field that contains the Siebel `RecordID`. The default value is `Id`.
InteractionField	M**		The name of the Business Component field that contains the Genesys `InteractionID`. There is no default value.
StatusField	M**		The name of the Business Component field that contains the status information. There is no default value.
SubStatusField	O		The name of the Business Component field that contains the substatus information. There is no default value.

Using the GplusMediaRoute Business Service

See “Using the GplusMediaRoute Business Service” on page 106 for more information.

The Applet Customization

See “The Applet Customization” on page 115 for more information.

Using the iWD Routing Component for Routing Siebel Work Items

The iWD Routing Component for Siebel can be used for routing any type of Siebel work item both in real-time and in background mode. The Media Routing Component provides basic functionality for Siebel work item routing, but customization is required to provide a graphical user interface (GUI). See the *Gplus Adapter 8.0 for Siebel CRM Deployment Guide* for general information on configuration of the Media Routing Component.

The `Group` buttons on the Communications toolbar work for all interaction types, such as `Logon/Logout`, `Accept`, and `Ready/NotReady`. However, you should create `Ready` and `NotReady` commands for custom media types and add them into the proper command groups, such as the `ReadyGroup` command group and the `NotReadyGroup` command group.

Creating Commands for Custom Media Types

Example 1 is a command sample.

Note: A media type is set with the prefix `@` in the `DeviceCommand` parameter and in the `FilterSpec` parameter.

Example 1

```
[Command:ReadyForSiebelSRGroup]
FilterSpec = "[$GetCommandStatus(@ServiceRequest@OpenMediaReady)] =
'Enabled'"
Hidden = "FALSE"
DeviceCommand = "@ServiceRequest@OpenMediaReady"
Description = "Set ready for SiebelSR media type"
Profile = " Gplus Universal Profile "
[Command:NotReadyForSiebelSR]
FilterSpec = "[$GetCommandStatus(@ServiceRequest@OpenMediaNotReady)] =
'Enabled'"
Hidden = "FALSE"
DeviceCommand = "@ServiceRequest@OpenMediaNotReady"
Description = "Set SiebelSR media type"
Profile = " Gplus Universal Profile "
```

Using the createtask Method to Send a createtask Request

To send a `createTask` request, you must use a `createTask` method of the `GplusMediaRouteiWD` business service. Example 2 is a sample command to route a service request. Change all `CHANGE_ME` values with appropriate values. Refer to the"The createTask Method" on page 130.

Example 2

```
[Command:SendRouteSR_iWD]
Description = "Route Service Request"
Title = "Route Service Request"
ServiceMethod = "GplusMediaRouteiWD.createTask"
Comments = "Send iWD.createTask request to route Siebel
ServiceRequest"
Hidden = "False"
AllViews = "False"
View = "Personal Service Request List View"
View = "All Service Request List View"
View = "Service Request Detail View"
Profile = "Gplus Universal Profile"
CmdData = "SendRouteCmdSR_iWD"
MenuPosition = "30.1"

[CmdData:SendRouteCmdSR_iWD]
RequiredField.SR Number = "?*"
ServiceParam.Subject = "{Abstract}"
ServiceParam.PrimaryOutboundDispatcherBS = "CHANGE_ME"
ServiceParam.BackupOutboundDispatcherBS = ""
ServiceParam.BusComp = "Service Request"
ServiceParam.BusObject = "Service Request"
ServiceParam.InteractionField = "Integration Id"
ServiceParam.RecIdField = "SR Number"
ServiceParam.SR_Type = "{SR Type}"
ServiceParam.ThirdPartyId = "{SR Number}"
ServiceParam.StatusField = "Status"
ServiceParam.SuccessStatus = "CHANGE_ME"
ServiceParam.FailedStatus = "CHANGE_ME"
ServiceParam.SubStatusField = "Sub-Status"
ServiceParam.SuccessSubStatus = "CHANGE_ME"
ServiceParam.FailedSubStatus = "CHANGE_ME"
ServiceParam.RefreshViewAfterUpdate = "Yes"
ServiceParam.BusinessValue = "CHANGE_ME"
ServiceParam.Priority = "CHANGE_ME"
; add any required createTask parameters here
Comments = ""
```

The `ServiceParam.ThirdPartyId` parameter should be set to the ID field for a Siebel work item record; it should be the same fields as in a proper event handler. Here the `SR Number` field value is used as a `ThirdPartyId`.

Note: he `InteractionField` parameter is set in Example 2, so it is possible to use the `Pull/Stop` functionality. As the `SuccessStatus` and `SuccessSubStatus` parameters are absent, the method does not update the record status. If you want to do a record status update, you can add the `SuccessStatus/SuccessSubStatus` and `FailedStatus/FailedSubStatus` parameters. See "The createTask Method" on page 130.

In Example 2, the `ServiceParam.Subject` is an optional parameter, and it is added as an `AttachedUserData` value. However, this value is used in a work item description text message. See the `itx_scdrv.xml` file in the Gplus Communication Server folder. For more information, see "Using the GplusMediaRouteiWD Business Service" on page 123.

Creating an Event Handler

Procedure: Creating an event handler

Purpose: To create an event handler for each custom media type to open a Siebel view for the routed work item.

Start of procedure

1. To accept a routed work item, the agent must click the `Accept` group button or the `Accept Multimedia Interaction` sub-button.
2. Create an event handler for the `OpenMediaAccepted` event for each custom media type to open a Siebel view for the routed work item.
3. Set the `QuerySpec` parameter to the value:

   ```
   FieldName='{ThirdPartyId}'
   ```

   where `FieldName` is the `ID` field for a Siebel work item.
4. Set the `SingleView` parameter to a proper Siebel view name. See Example 3 and Example 4:

Example 3

```
[EventHandler:OpenMediaSelectedSR]
    Filter.MediaType = "ServiceRequest"
    Profile = "Gplus Universal Profile"
    Comments = "EventHandler samples for Siebel work items
```

```
        routing"
    Order = "50"
    Response = "OpenSiebelViewSR"
    DeviceEvent = "OpenMediaSelected"

[EventResponse:OpenSiebelViewSR]
    QueryBusComp = "Service Request"
    QueryBusObj = "Service Request"
    QuerySpec = "SR Number='{ThirdPartyId}'"
    SingleView = "Service Request Detail View"
    Comments = "EventResponse samples for Siebel work items
        routing"
```

If you want to update a record status and/or assign it to the agent when an agent accepts the interaction, you should create an event handler as shown in Example 4:

Example 4

```
[EventHandler:OpenMediaAcceptedSR]
    Filter.ThirdPartyId = "?*"
    Filter.MediaType = "ServiceRequest"
    DeviceEvent = "OpenMediaAccepted"
    Profile = "Gplus Universal Profile"
    Response = "EventResponseAcceptSR"
    Order = "50"

    [EventResponse:EventResponseAcceptSR]
        QueryBusComp = "Service Request"
        QueryBusObj = "Service Request"
        Log = "EventLogAcceptSR"

    [EventLog:EventLogAcceptSR]
        BusComp = "Service Request"
        BusObj = "Service Request"
        LogField.Owner = "{@UserName}"
        LogField.Status = "CHANGE_ME"
        LogField.Sub-Status = "CHANGE_ME"
        QuerySpec = "SR Number='{ThirdPartyId}'"
```

End of procedure

Next Steps

- There are no further steps.

You have to provide a command to mark a work item as done. For this you may configure the provided `MarkWorkItemDone` command to support your custom types, such as adding the media types into the `FilterMediaType` parameter value, which contains a list of media types separated by commas. As an alternative, you may customize a button on a work item view or add a custom button on the Communications toolbar to invoke a `MarkDone` command. Refer to the section "The Applet Customization" on page 115.

The provided out-of-the-box sample shows how to route Siebel Service Requests in real-time mode and how to route Service Orders in background mode. It uses the `ServiceRequest` media type for Siebel Service Requests and the `ServiceOrder` media type for Service Orders.

Procedure: Using the provided sample to route Siebel service request

Purpose: To use the provided sample to route Siebel Service Requests in real-time mode and how to route Service Orders in background mode.

Start of procedure

1. Remove the comment marks in the `GenComm_universal.def` (appropriate places have the following comment: "`Use this command in couple with Gplus iWD Routing for Siebel CRM`") file before importing a configuration and performing the above-mentioned actions.
2. Create a custom media type in the Genesys environment and then add in the `Channel String` parameter of the `Gplus Universal Profile` profile driver.
3. Set the proper values for the command and events used.

End of procedure

Next Steps

- There are no further steps.

For more information about the iWD Routing Component deployment and configuration, refer to Chapter 8 in the *Gplus Adapter 8.0 for Siebel CRM Deployment Guide*.

Chapter

4 Using Siebel Data from the Genesys Universal Routing Solution

It may be beneficial to use the data stored in Siebel CRM or to invoke some Siebel functionality from the Genesys Universal Routing solution. This can be achieved through the Web Service (Simple Object Access Protocol—SOAP) interface. On a high level, the data access should be represented as a Siebel business service and exposed as an inbound web service. On the Genesys side, the web service strategy-building object should be used to invoke the business service from a routing strategy.

This chapter describes use of Siebel data from the Genesys Universal Routing solution in the following sections:

Checking the Inbound Web Service

Procedure:
Managing the Siebel inbound web services on a Siebel client

Purpose: To administer Siebel inbound web services on the Siebel client

Start of procedure

1. Navigate to `View` > `Site Map` > `Administration - Web Services` > `Inbound Web Services`. See Figure 3.
2. Select a web service. (This chapter uses the standard Siebel Contact web service as an example.)

   For more information about how to create your own Inbound Web Service, refer to the Siebel documentation.

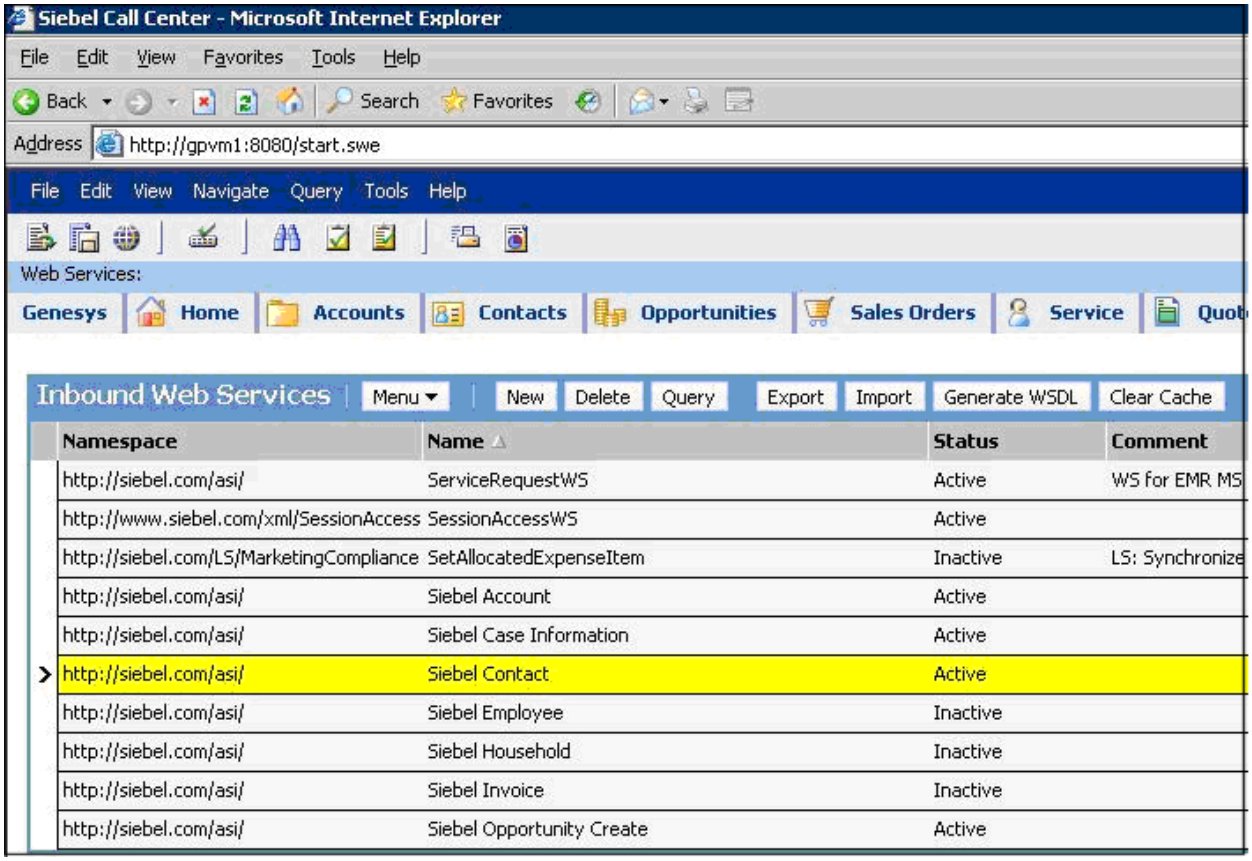

Figure 3: The Siebel Inbound Web Services Administration View

End of procedure

Next Steps

- There are no further steps.

Using the Web Service

On the Genesys side, include the Web Service object into an appropriate routing strategy to perform SOAP requests to the Siebel database. See Figure 4.

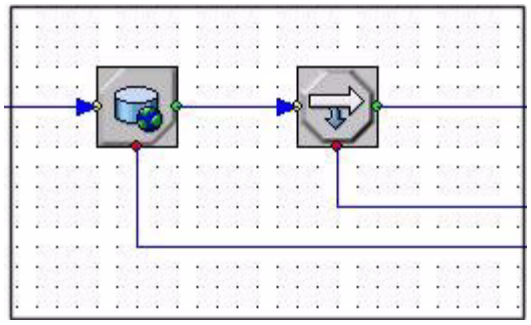

Figure 4: Example of Using the Web Service Object in a Routing Strategy

The `Web Service` object has the following properties (see Figure 5 on page 158):

- The `Web Service URL` contains the URL for the appropriate web service.

  For this example, the URL has the following structure:

  ```
  http://<webserver>/eai_<lang>/start.swe?SWEExtSource=WebService&SWEExtCmd=Execute&UserName=<UserName>&Password=<Password>
  ```

  You can get this URL from the `Address` field of the `Service Port` applet of the `Inbound Web Services Administration` view
- The `Method name` contains the SOAP method name to invoke. It has the following structure:

  ```
  <namespase variable name>:<SOAP method name>
  ```

  You can find the valid method names in the `Operations` applet of the Inbound Web Services Administration view.
- The `Method namespace` contains the namespace for the SOAP request. It has the following structure:

  ```
  <namespase variable name>=< namespace>
  ```
- The `SOAPaction` contains the SOAP action. It has the following structure:

  ```
  rpc/< SOAP method name >
  ```
- `The Request parameters` contains the key-value pairs that correspond to the SOAP method input parameters.

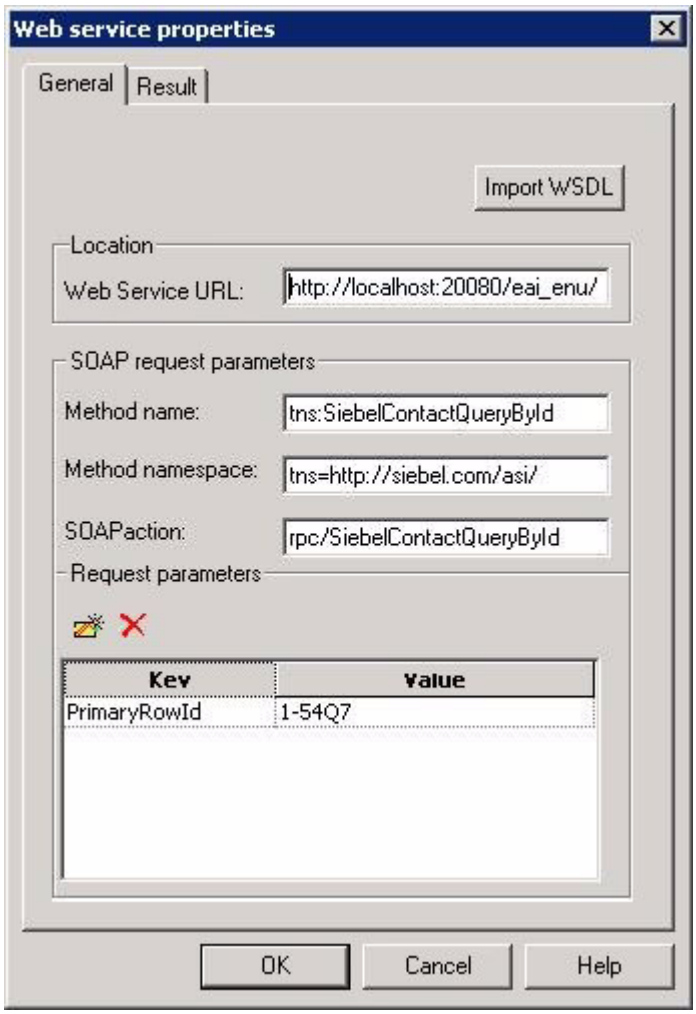

Figure 5: Web Service Object Properties - General Tab

To get the method's parameter list, refer to the Siebel Tools `Business Service` screen. You can find appropriate business service names in the `Business Service` field of the `Service Port` applet of the `Inbound Web Services Administration` view.

You can assign the *result* of the SOAP request, for example, to a strategy local variable. (See Figure 6 on page 159.)

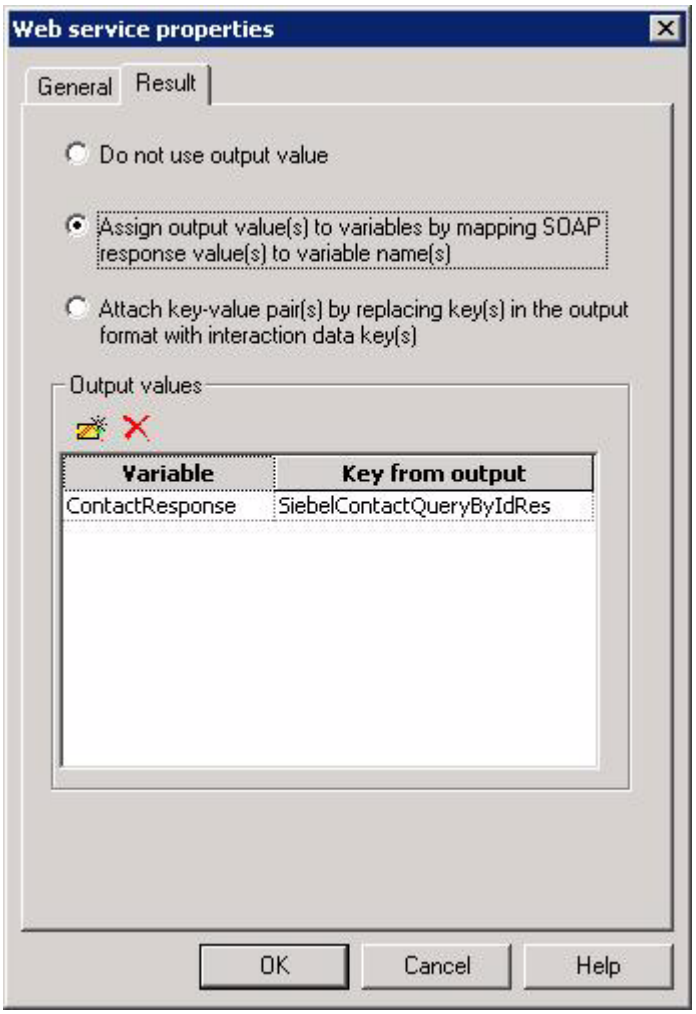

Figure 6: Web Service Object Properties - Result Tab

Reference to the SOAP response generally depends on the requested method output variables type—for example, the reference to the `EmailAddress` field is:

```
SiebelContactQueryByIdResponse.SiebelMessage.ListOfContactInterface.Contact.Contact.EmailAddress
```

For more information related to the `Web Service` object functionality of Genesys Universal Routing, refer to the Universal Routing documentation.

You can operate with the SOAP request results using an `If expression` strategy object—for example, see Figure 4 on page 157 and Figure 7 on page 160.

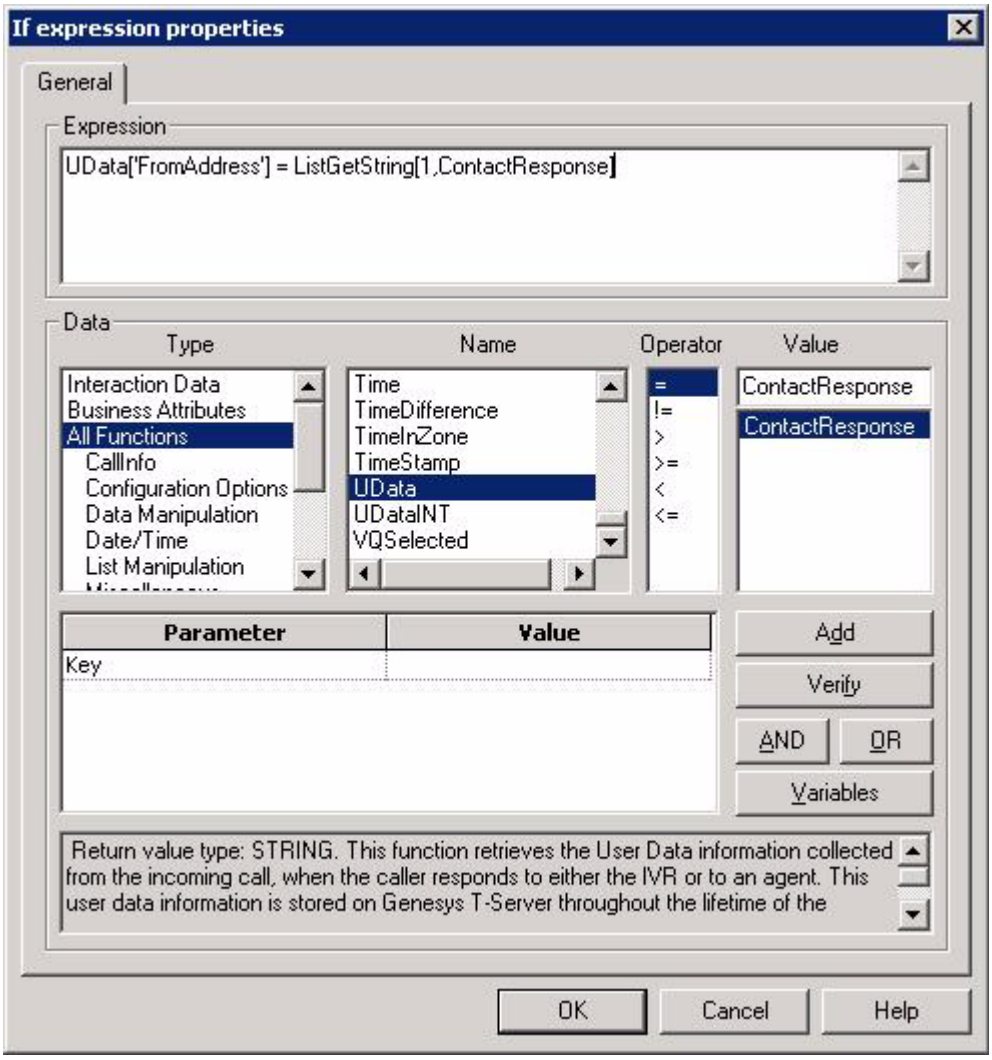

Figure 7: "If" Expression Object Parameters

For example, you can compare what is assigned in a `Web Service` object variable with other data.

Notes: If the Siebel `Web Service` method returns the `SiebelMessage` value (an Integration object), the result key value will be a list instead of a base typed value. Use the `ListGetXXX[]` method to retrieve the base typed values from this list.

Currently, the Web Service strategy building block does not support input/output parameters of arbitrary complexity. If it is necessary to use such a business service, it should be wrapped up into another business service with plain parameters. For general information about the Web Service strategy building block, please refer to the Genesys Universal Routing documentation.

Genesys

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

G*plus* Adapter for Siebel CRM

- *Gplus* Adapter 8.0 for Siebel CRM *User's Guide*. Explains how to use the Adapter in your contact center environment. Introduces the G*plus* Adapter and highlights new features in the current release.
- *Gplus* Adapter 8.0 for Siebel CRM *Deployment Guide*. This guide lists system requirements and describes how to install and configure the G*plus* Adapter.

Genesys

- *Genesys Technical Publications Glossary,* which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which provides documented migration strategies for Genesys product releases. Contact Genesys Customer Care for more information.
- Release Notes and Product Advisories for this product, which are available on the Genesys Customer Care website at http://genesys.com/customer-care.

Information about supported hardware and third-party software is available on the Genesys Customer Care website in the following documents:

- *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*

Consult these additional resources as necessary:

- *Genesys Hardware Sizing Guide,* which provides information about Genesys hardware sizing guidelines for the Genesys 8.0 releases.

- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures relevant to the Genesys licensing system.
- *Genesys Database Sizing Estimator 8.x Worksheets,* which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System-Level Documents on the Genesys Documentation website.

Genesys product documentation is available on the:

- Genesys Customer Care website.
- Genesys Documentation website.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.com.

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

80fr_ref_06-2008_v8.0.001.00

You will need this number when you are talking with Genesys Customer Care about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 36 describes and illustrates the type conventions that are used in this document.

Table 36: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 164).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x + 1 = 7$ where x stands for . . .

Table 36: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like `teletype` or `typewriter text`)	All programming identifiers and GUI elements. This convention includes: • The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. • The values of options. • Logical arguments and command syntax. • Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages in `EventError` events. If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Index

Symbols

A

B

C

D

E

F

G

I

L

M

O

P

R

S

T

U

V

W