



***Gplus* Adapter 7.5**

Gplus Adapter for Siebel CRM

Developer's Guide



The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of Genesys Telecommunications Laboratories, Inc.

Copyright © 2001–2007 Genesys Telecommunications Laboratories, Inc. All rights reserved.

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library CD, 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, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. 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.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-118-974-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-5649-6871	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the [Genesys Technical Support Guide](#) for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the [Genesys 7 Licensing Guide](#).

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 75gp_dev_slcrm_08-2007_v7.5.001.00



Table of Contents

Preface	5
Intended Audience	6
Usage Guidelines	6
Chapter Summaries	8
Document Conventions	9
Related Resources	11
Making Comments on This Document	11
Chapter 1	Campaign Synchronization Component Overview 13
Campaign Synchronization Data Flow	13
Configuring the List Import Functionality	14
Genesys Campaign Synchronization Business Service	15
Default Scenarios for Campaign Synchronization	16
Synchronization Summary Usage	18
Chapter 2	Campaign Synchronization Data Flow 19
Exporting Siebel Campaign Lists to Genesys	19
The Inbound XML Schema	26
Campaign Synchronization Request Types	29
CampaignInfo Requests and Attributes	29
ListInfo Requests and Attributes	31
RecordInfo Requests and Attributes	34
Campaign Synchronization Response	34
The Outbound XML Schema	36
Chapter 3	Configuring the List Import Functionality 39
Siebel Configuration Guidelines	39
Format Configuration Guidelines	40
Style Sheet Configuration Guidelines	43
Style Sheet Configuration Examples	45
Example 1	45
Example 2	46

Chapter 4	Using the Genesys Campaign Synchronization Business Service .	49
	Method Usage Guidelines	50
	Method Descriptions	50
	PreSubmitRequests	50
	SubmitRequests	51
	ExportCampaign	51
	DeleteCampaign	51
	ExportLists	52
	AppendList	52
	DeleteLists	53
	AssignLists	53
	ExportCampaignWithLists	53
	DeleteCampaignWithLists	54
	InsertListRecords	54
	UpdateListRecords	55
	DeleteListRecords	55
	InsertDNCListRecord	55
	DeleteDNCListRecord (obsolete)	56
	Script Example	56
Chapter 5	Synchronization Summary Usage	59
	Synchronization Summary Format	59
	Using Synchronization Summary	61
Chapter 6	Media Routing Component Customization	65
	Overview	65
	Using the GplusMediaRoute Business Service	66
	GetTopWorkItem Method	66
	MarkWorkItemDone and Special MarkDone Methods	66
	PullInteraction Method	69
	StopWorkItem Method	70
	Route Method	71
	Applet Customization	74
	Using the Media Routing Component for Routing Siebel Work Items	74
Chapter 7	Using Siebel Data from the Genesys Universal Routing Solution...	79
	Checking the Inbound Web Service	79
	Using the Web Service in Genesys Universal Routing	80
Index	85



Preface

Welcome to the *Gplus Adapter 7.5 for Siebel CRM Developer's Guide*. In general, this document addresses only the interactions of Genesys *Gplus* Adapter components with other Genesys systems and products. Developers who are using Siebel application development tools and services to implement the *Gplus* Adapter should look in the Siebel documentation set for information.

The *Gplus Adapter 7.5 for Siebel CRM Deployment Guide* may contain information useful for developers who need to customize the *Gplus* Adapter 7.5 for Siebel CRM. You should have ready access to this document, if only to understand the standard *Gplus* 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 Campaign Synchronization Component's customization options.
- An explanation of Campaign Synchronization data flow.
- A summary of the scripts provided in the Campaign Synchronization Component's default implementation.
- A listing of methods exposed by the Campaign Synchronization Component's business service, including descriptions of the methods' purposes, required calling order, and required parameters.
- A script example that illustrates how to call the campaign synchronization methods.
- Style sheet customization examples that illustrate how to modify the XSL file to set the field values conversion.
- An overview of the Media Routing Component's interface and customization options.

This preface introduces basic concepts, prerequisites, and typographic conventions that underlie the guide's instructions for customizing the Adapter's behavior. The chapter contains the following sections:

- [Intended Audience, page 6](#)
- [Usage Guidelines, page 6](#)
- [Chapter Summaries, page 8](#)
- [Document Conventions, page 9](#)

- [Related Resources, page 11](#)
- [Making Comments on This Document, page 11](#)

Intended Audience

This guide is intended for developers who will customize the behavior of the *Gplus* 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 7.0, 7.1, 7.2, 7.5
- Outbound Contact Server 7.5

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 Genesys software implementations.
- Server-side integration between Genesys software and third-party software.
- Creation of a specialized client application specific to customer needs.

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

The following Use Conditions 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. 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 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 participant and Genesys databases, unless it can be assured that data modifications propagate all copies within the time required by 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 design and implementation guidelines and restrictions described in the *Genesys Developer Program Guide* and Genesys software documentation. For example:
 - a. The integration must use only published interfaces to access Genesys data.
 - b. The integration shall not modify data in Genesys database tables directly using SQL.
 - c. The integration shall not introduce database triggers or stored procedures that operate on Genesys database tables.

Any schema extension to Genesys database tables must be carried out using Genesys developer software through 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 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.

Chapter Summaries

In addition to this preface, the *Gplus Adapter for Siebel CRM Developer's Guide* describes the Campaign Synchronization Component's customization options, and provides examples of how to use those options. The guide contains the following chapters:

- Chapter 1, "Campaign Synchronization Component Overview," on [page 13](#), describes the Campaign Synchronization Component's customization options and default implementation.
- Chapter 2, "Campaign Synchronization Data Flow," on [page 19](#), provides a brief explanation of data exchange between the Siebel and Genesys portions of the Adapter.
- Chapter 3, "Configuring the List Import Functionality," on [page 39](#), describes how to customize List Import functionality, add additional custom fields, and modify imported field values.
- Chapter 4, "Using the Genesys Campaign Synchronization Business Service," on [page 49](#), lists and describes the campaign synchronization methods exposed by the Campaign Synchronization Component. It also provides an example of Siebel eScript that illustrates how to call several of these methods.
- Chapter 5, "Synchronization Summary Usage," on [page 59](#), provides guidelines for using the Genesys Synch Summary Business Service functionality.
- Chapter 6, "Media Routing Component Customization," on [page 65](#), describes the `GplusMediaRoute` Business Service, applet customization, and using the Media Routing Component to route Siebel work items.
- Chapter 7, "Using Siebel Data from the Genesys Universal Routing Solution," on [page 79](#), describes use of Siebel data from the Genesys Universal Routing solution.

Document Conventions

This guide uses the following document conventions:

Words and Terminology

Throughout this document, the Voice, Multimedia, and Media Routing components of the *Gplus* Adapter are categorized as “driver-based components,” whereas the Configuration Synchronization, Campaign Synchronization, Communication Server, and UCS Gateway components of the *Gplus* Adapter are called “server-based components.”

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:

75gp_dev_slcrm_08-2007_v7.5.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Type Styles

Italic

In this document, italic is used for the titles of documents, when a term is being defined, for emphasis, and for mathematical variables.

- Examples**
- Please consult the *Genesys 7 Migration Guide* for more information.
 - *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
 - Do *not* use this value for this option.
 - The formula, $x + 1 = 7$ where x stands for . . .

Monospace Font

A monospace font, which is shown in the following examples, is used for:

- 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; and code samples.

- Examples**
- Select the Show variables on screen check box.

- Click the `Summation` button.
- In the `Properties` dialog box, enter the value for the host server in your environment.
- In the `Operand` text box, enter your formula.
- Click `OK` to exit the `Properties` dialog box.
- The following table presents the complete set of error messages T-Server® distributes 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.
- For any text the user must manually enter during a configuration or installation procedure:

Example

- Enter `exit` at the command line.

Information About Screen Captures Used in This Document

Screen captures taken from the product GUI (graphical user interface) and used in this document may contain minor errors in spelling, capitalization, or grammar. 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 an option in the Siebel user interface contains a spelling error, then this document may use the name exactly as it appears in that Siebel user interface; such errors are not necessarily corrected in any accompanying text.

Use of Square Brackets

In any logical arguments, commands, and programming syntax presented in this document, square brackets are used to indicate that a particular parametric value is optional. That is, the value is not required to resolve a command, argument, or programming syntax. The customer/user decides whether to supply a value and what that value is. Here is a sample:

```
smcp_server -host [/flags]s
```

Use of Angle Brackets

Angle brackets are used to indicate that a value in a logical argument, command, or programming syntax is required, but that the user must supply the data for the value. Because the value is specific to an individual enterprise—for example, DNs or port numbers—the program cannot predict (that is, program in) what the value is. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

For information about *Gplus* Adapter for Siebel CRM not covered in this guide, consult these documents:

- *Gplus Adapter 7.5 for Siebel CRM Deployment Guide*. Lists system requirements and describes how to install and configure *Gplus* Adapter.
- *Gplus Adapter 7.5 for Siebel CRM User's Guide*. Provides examples of how to use *Gplus* Adapter in your contact center environment.
- *Gplus Adapter 7.5 for Siebel CRM Release Notes*. Includes known limitations and restrictions. Note that there is one separate set of Release Notes for each major component; that is, one set of release notes for Voice, another for Multimedia, another for Configuration Synchronization, and so on.
- *Genesys Migration Guide*. This is provided on the Genesys Documentation Library CD. It describes a migration strategy for Genesys products. When you are ready to migrate to the latest Genesys software, please refer to the applicable portion of the *Genesys Migration Guide*, or contact Genesys Technical Support for additional information.
- Your Siebel software documentation, notably the *Siebel Bookshelf*.

Genesys product documentation is available on the:

- Genesys Documentation Library CD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.
- Genesys Technical Support website at <http://genesyslab.com/support>.

Making Comments on This Document

If you especially like or dislike anything about this document, please feel free to e-mail your comments to Techpubs.webadmin@genesyslab.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 information in this document only and to the way in which the information is presented. Speak to Genesys Technical Support 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.



Chapter

1

Campaign Synchronization Component Overview

This chapter provides a developer's overview of *Gplus* Adapter's Campaign Synchronization Component, focusing on the customization options and how to use them.

Each of the topics in this chapter describes one aspect of the component that you can use to customize campaign synchronization between the Siebel and Genesys environments:

- [Campaign Synchronization Data Flow, page 13](#)
- [Configuring the List Import Functionality, page 14](#)
- [Genesys Campaign Synchronization Business Service, page 15](#)
- [Default Scenarios for Campaign Synchronization, page 16](#)
- [Synchronization Summary Usage, page 18](#)

Other chapters of this guide cover other Adapter components. For a functional overview of the complete *Gplus* Adapter and its integration with Siebel software, see the *Gplus Adapter 7.5 for Siebel CRM User's Guide*.

For a detailed description of the Siebel EAI architecture, see your Siebel documentation—especially *Integration Platform Technologies: Siebel eBusiness Application Integration Volume II*.

Campaign Synchronization Data Flow

Chapter 2, “Campaign Synchronization Data Flow,” on [page 19](#) provides a brief explanation of data interaction between the Siebel and Genesys portions of the Adapter. It describes all types of Adapter messages with examples. The chapter also includes a formal definition of Inbound/Outbound XML messages as XML schemas.

Configuring the List Import Functionality

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

- The Genesys Configuration Layer format object that is specified by the `format application` option.
- The Siebel Integration object - Genesys - Campaign List Contact.

The Campaign Synchronization Component uses the format object to create Genesys Calling List tables.

The default format used by the Campaign Synchronization Component is `GplusCampSynch`. You create this format according to instructions provided in the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide*. The Genesys-specific Siebel Integration object `Genesys - Campaign List Contact` contains the default mapping of Siebel fields to Genesys fields. This mapping is shown in [Table 1](#).

Table 1: Default Mapping of Siebel to Genesys Fields

Siebel Field	Genesys Field
CampaignListContact.HomePhone, CampaignListContact.WorkPhone, CampaignListContact.ProspectHomePhone, CampaignListContact.ProspectWorkPhone,	contact_info, contact_info_type, where contact_info_type = 1 for home phones, and contact_info_type = 2 for work phones
CampaignListContact.TimeZoneName, CampaignListContact.ProspectTimeZone	tz_dbid
CampaignListContact.CampaignId	crm_campaign_id
CampaignListContact.Id	crm_camp_con_id
CampaignListContact.ContactId, CampaignListContact.ProspectId	crm_contact_id

You can use both the default format and the Genesys-specific Siebel Integration Object as the starting point to create custom list import formats.

Using the `format` option, you can specify a different format at any time. The mapping of Siebel to Genesys fields—which is provided by the corresponding Genesys-specific `Genesys - Campaign List Contact` Siebel Integration Object—should cover all specified formats and user-defined fields.

For more information, see Chapter 3, “Configuring the List Import Functionality” on [page 39](#).

Genesys Campaign Synchronization Business Service

The Genesys Campaign Synchronization Siebel Business Service, which installs with the Campaign Synchronization Component, encapsulates the component’s ability to synchronize campaign data from the Siebel to the Genesys environment. This Business Service exposes a set of methods used to perform the following tasks on campaign data in the Genesys environment:

- Export or delete Campaigns.
- Export, append, or delete Lists.
- Assign Lists to Campaigns.
- Insert, update, or delete List records.
- Insert records to the Genesys “Do Not Call” List.

You import the Genesys Campaign Synchronization Business Service into the Siebel Repository when you deploy the Campaign Synchronization Component.

You can invoke the Genesys Campaign Synchronization Business Service as you would invoke any other Siebel Business Service. For example, you can invoke it from the:

- Siebel eScript environment.
- Siebel VB environment.
- Siebel Workflow environment.

For descriptions of the Genesys Campaign Synchronization Business Service’s methods and their arguments, see Chapter 4, “Using the Genesys Campaign Synchronization Business Service” on [page 49](#). For general information about Siebel Business Services and their role in Siebel applications, see your Siebel documentation.

Warning! If Genesys Outbound Contact is running when you update the Genesys environment using the Genesys Campaign Synchronization Business Service, the Outbound Contact is not notified about the updates (excluding “DoNotCall” records, which update dynamically). To avoid inconsistency of information between the Genesys and Siebel environments, do not update data for campaigns which are being executed by Genesys Outbound Contact at this moment. In other words, call the methods of the Genesys Campaign Synchronization Business Service only for those campaigns which are not being executed by Genesys Outbound Contact yet, or when Genesys Outbound Contact is *not* running.

Default Scenarios for Campaign Synchronization

When installed, the Campaign Synchronization Component supports a number of default scenarios for campaign synchronization between the Genesys and Siebel environments. Each scenario corresponds to a particular campaign synchronization task. These scenarios are encapsulated by the Genesys Campaign Synchronization Business Service.

The Campaign Synchronization Component uses a set of Siebel RunTime Events to invoke the campaign synchronization scenarios. [Table 2](#) lists RunTime Events corresponding to the affected Business Components.

Table 2: Business Components and Their RunTime Events

Business Component	Business Component Event
Campaign	BusComp_PreSetFieldValue
	BusComp_SetFieldValue
	BusComp_CopyRecord
	BusComp_NewRecord
	BusComp_PreDeleteRecord
	BusComp_DeleteRecord
	BusComp_PreWriteRecord
	BusComp_WriteRecord
Campaign List Contact	BusComp_CopyRecord

Table 2: Business Components and Their RunTime Events (Continued)

Business Component	Business Component Event
	BusComp_NewRecord
	BusComp_PreDeleteRecord
	BusComp_DeleteRecord
	BusComp_PreWriteRecord
	BusComp_WriteRecord
Campaign Lists (Siebel 7.5.3 only)	BusComp_Associate
	BusComp_PreDeleteRecord
	BusComp_DeleteRecord
Campaign Load Wave (Siebel 7.7/7.8/8.0)	BusComp_PreWriteRecord
	BusComp_WriteRecord
Consumer, Contact, Employee, List Mgmt Prospective Contact, Person, User	BusComp_PreSetFieldValue
	BusComp_SetFieldValue
	BusComp_PreDeleteRecord
	BusComp_DeleteRecord
	BusComp_PreWriteRecord
	BusComp_WriteRecord
Position	BusComp_PreAssociate
	BusComp_Associate
	BusComp_PreDeleteRecord
	BusComp_DeleteRecord

You can customize or extend the default campaign synchronization scenarios by modifying the Genesys Campaign Synchronization Business Service and the RunTime Events referenced in [Table 2](#). Or, you can use this Business Service and these RunTime Events as starting points in creating your own campaign synchronization scenarios.

Synchronization Summary Usage

The Genesys Synch Summary Siebel Business Component, which installs with the Campaign Synchronization Component, is used to collect in the Siebel environment the summary information about results of data synchronization requests sent from Siebel to Genesys.

The Genesys Campaign Synchronization Adapter, after executing a data synchronization request in the Genesys environment, sends synchronization summary information about the request back to Siebel in the corresponding response. The summary information is then stored in a Genesys Synch Summary Siebel Business Component.

The Genesys Synch Summary Siebel Business Component can be used to create one or more Siebel Client views to show synchronization summary results to the user.

Customers who want synchronization summary information should change the value of the synchSummary option of the Genesys Campaign Synchronization application object in Genesys Configuration Manager Environment to true.

For more information, see Chapter 5, “Using Synchronization Summary” on [page 61](#).



Chapter

2

Campaign Synchronization Data Flow

This chapter provides a brief explanation of data interaction between the Siebel and Genesys portions of the Adapter. It describes all types of Adapter messages with examples, and also includes a formal definition of Inbound/Outbound XML messages as XML schemas.

The chapter contains the following sections:

- [Exporting Siebel Campaign Lists to Genesys, page 19](#)
- [The Inbound XML Schema, page 26](#)
- [Campaign Synchronization Request Types, page 29](#)
- [Campaign Synchronization Response, page 34](#)
- [The Outbound XML Schema, page 36](#)

Exporting Siebel Campaign Lists to Genesys

The main element of the data flow is the Campaign List (the Calling List in Genesys). On the Siebel side, all Campaign Lists are contained within one database table; Campaign List Contact (S_CAMP_CON). On the Genesys side, all corresponding Calling Lists are contained within *different* tables. In general, all Calling Lists exported from Siebel have the same table format contained in the Configuration Layer Environment (CLE), which the Adapter uses for table creation and update. Refer to Figure 1 on [page 20](#) for a diagram depicting export of Siebel Campaign Lists to Genesys.

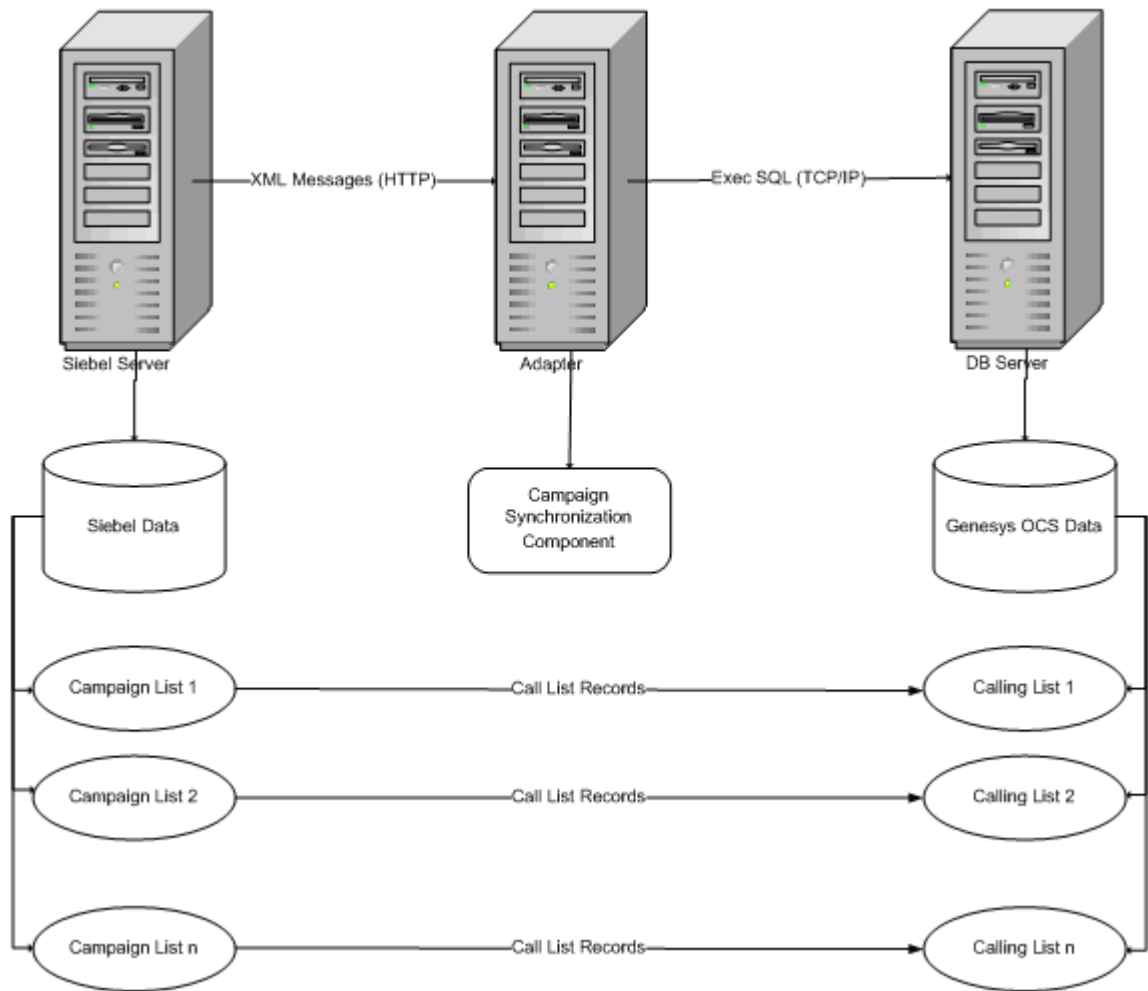


Figure 1: Exporting Siebel Campaign Lists to Genesys

The major difference between a Siebel Campaign List and a Genesys Calling List is that one record in Siebel transfers to one or many records in Genesys depending on how many phone relative fields (`contact_info`) are exported from one Siebel record.

For example, assume we have a Siebel Campaign List containing a record as shown below in Table 3 on [page 21](#):

Table 3: Sample Siebel Campaign List (One Record)

ID	Home Phone	Work Phone	Mobile Phone	Custom Field1
1-AABC	111-222-3301	222-333-4401	333-444-5501	Anything

After synchronization with Genesys, this Siebel record transforms into *three* records in a Genesys Calling List table as shown below in [Table 4](#):

Table 4: Sample Campaign List Transformed into Genesys Calling List

chain_id	chain_n	crm_camp_con_id	contact_info	contact_info_type	cd_field1	cd_field2
11	0	1-AABC	111-222-3301	1	Anything	Home
11	1	1-AABC	222-333-4401	2	Anything	Other
11	2	1-AABC	333-444-5501	4	Anything	Other

In this example we assume that the customer has *two* custom fields (`cd_field1` and `cd_field2`) in Genesys Calling List records and added “Mobile Phone” to the synchronization list. We also assume that the Genesys field `cd_field1` corresponds to the Siebel field `Custom Field1` and the value of `cd_field2` should equal “Home” for home phones or “Other” for other phone types.

The detailed data flow is shown in Figure 2 on [page 22](#) below.

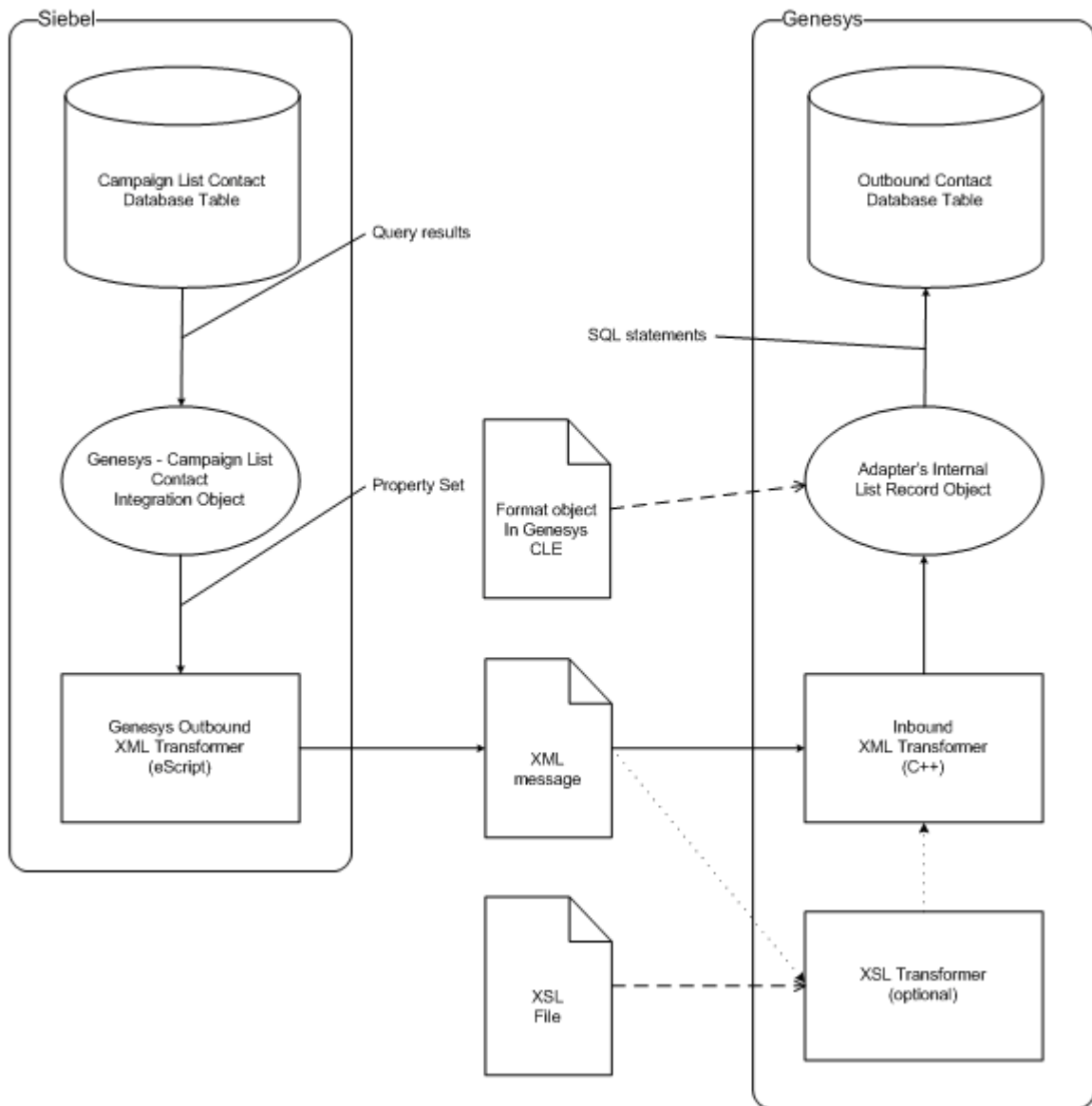


Figure 2: Sample Scenario Data Flow

The Adapter makes a query of the Siebel database by using the Siebel Integration Object Genesys - Campaign List Contact. This Integration Object is a key element for List Import customization (for more information see Chapter 3, “Configuring the List Import Functionality,” on [page 39](#)). It defines which Siebel Campaign List fields to use and how to transform given query results to the appropriate Siebel Property Set (mapping between Siebel field names and Genesys field names).

In our example, we obtained a Property Set as shown below:

```
<PropertySet ...>
  ...
  <ContactInfo
    crm_camp_con_id="1-AABC"
    phone_1="1112223301"
    phone_2="2223334401"
    phone_4="3334445501"
    ...
    cd_field1="Anything">
  </ContactInfo>
  ...
</PropertySet>
```

The result of data transformation on the Siebel side is an XML message. In our example, the transformation resulted in an XML-message as shown below:

```
<ContactInfo
  crm_camp_con_id="1-AABC"
  ...
  cd_field1="Anything">
  <Record
    contact_info="1112223301"
    contact_info_type="1"
  >
  </Record>
  <Record
    contact_info="2223334401"
    contact_info_type="2"
  >
  </Record>
  <Record
    contact_info="3334445501"
    contact_info_type="4"
  >
  </Record>
  ...
</ContactInfo>
```

The key element of the Adapter's XML message is <ContactInfo> which is generated from the ContactInfo Property Set. The Adapter generates for each attribute phone_X (where X is the value of contact_info_type) a new element <Record> with two attributes: contact_info and contact_info_type. This

element is intended to deliver specific field values for any record in a chain. Attributes (records' fields) with common values for the entire chain may be put in the element `<ContactInfo>` (`cd_field1` in our example). The names of all attributes in `<Record>` and `<ContactInfo>` should be the same as the corresponding names of the Genesys Outbound Calling List fields (the Format fields in the Genesys CLE).

The attribute `crm_camp_con_id` is a unique key for the element `<ContactInfo>` (a chain in OCS). For identification of the record in a chain there are three attributes that may be used: `chain_n`, `contact_info_type`, or `contact_info`. The attribute `chain_n` is always used for identification when it is included in `<Record>`. If `chain_n` is not included, Genesys uses `contact_info_type` if it is included in `<Record>`, or `contact_info` otherwise.

The last customization point is an XSL Transformer (on the Genesys side of the Adapter). We can create an XSL file which may be used by the Adapter as rules for transformation of all inbound messages (see more details in "Style Sheet Configuration Guidelines" on [page 43](#) in [Chapter 3](#)).

In our example we need to set up different values for the field `cd_field2` depending on the phone type. After the XSL transformation, the message should contain the `<ContactInfo>` element as shown below:

```
<ContactInfo
  crm_camp_con_id="1-AABC"
  ...
  cd_field1="Anything">
    <Record
      contact_info="1112223301"
      contact_info_type="1"
      cd_field2="Home"
    >
    </Record>
    <Record
      contact_info="2223334401"
      contact_info_type="2"
      cd_field2="Other"
    >
    </Record>
    <Record
      contact_info="3334445501"
      contact_info_type="4"
      cd_field2="Other"
    >
    </Record>
    ...
</ContactInfo>
```


The example of the XSL file which can make such a transformation is shown below:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="windows-1252"/>
  <xsl:template match="/" | @* | node() ">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>
  <xsl:template match="@contact_info_type">
    <xsl:attribute name="contact_info_type"><xsl:value-of
      select="."/></xsl:attribute>
    <xsl:attribute name="cd_field2">
      <xsl:choose>
        <xsl:when test=". = '1'">
          <xsl:value-of select="Home"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="Other"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
  </xsl:template>
</xsl:stylesheet>
```

After this point, customization is impossible. The Adapter transforms XML messages to internal List Record objects based on the assigned Format object in Genesys CLE. Then, the Adapter generates SQL statements and sends them to the database used by OCS.

The structure of all possible inbound messages has a formal definition - the Inbound XML Schema. You can use this schema for developing your own application instead of using the Siebel part of the Adapter.

The Inbound XML Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Gplus Adapter 7.5. Campaign Synchronization.Schema of Inbound Messages -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definition of attributes -->
  <xs:attribute name="Name" type="xs:string" />
  <!-- the request type -->
  <xs:attribute name="Id" type="xs:string" />
  <xs:attribute name="TenantId" type="xs:string" />
  <xs:attribute name="AdminLoginName" type="xs:string" />
  <xs:attribute name="CampaignName" type="xs:string" />
  <xs:attribute name="crm_camp_con_id" type="xs:string" />
  <xs:attribute name="crm_campaign_id" type="xs:string" />
  <xs:attribute name="contact_info" type="xs:string" />
  <xs:attribute name="contact_info_type" type="xs:string" />
  <xs:attribute name="FirstPage" type="xs:string" />
  <xs:attribute name="LastPage" type="xs:string" />
  <!-- *** -->
  <!-- definition of elements -->
  <xs:element name="Record">
    <xs:complexType>
      <xs:attribute ref="contact_info" use="required" />
      <xs:attribute ref="contact_info_type" use="required" />
      <!-- The name of all additional attributes should be matched to valid
      field names from OCS table format. These fields may have specific value for
      each record in a chain -->
      <xs:anyAttribute />
    </xs:complexType>
  </xs:element>
  <!-- used in DoNotCall request only -->
  <xs:element name="CustomerId" type="xs:string" />
  <xs:element name="Phone" type="xs:string" />
  <xs:element name="RecordInfo">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="CustomerId" />
        <xs:element ref="Phone" />
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <!-- *** -->
  <xs:element name="ContactInfo">
    <xs:complexType>
      <xs:sequence>

```

```

        <xs:element ref="Record" minOccurs="1" />
    </xs:sequence>
    <xs:attribute ref="crm_campaign_id" use="optional" />
    <xs:attribute ref="crm_camp_con_id" use="required" />
<!-- The name of all additional attributes should be matched to valid field
names from OCS table format. These fields have common values for all records in
a chain -->
    <xs:anyAttribute />
</xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="ListOfContactInfo">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="ContactInfo" minOccurs="1" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="ListInfo">
    <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="1">
            <xs:element ref="ListOfContactInfo" />
        </xs:sequence>
        <xs:attribute ref="TenantId" use="required" />
        <xs:attribute ref="AdminLoginName" use="required" />
        <xs:attribute ref="CampaignName" use="optional" />
        <xs:attribute ref="Name" use="required" />
        <xs:attribute ref="FirstPage" use="optional" />
        <xs:attribute ref="LastPage" use="optional" />
        <xs:anyAttribute />
    </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="List">
    <xs:complexType>
        <xs:attribute ref="CampaignName" use="optional" />
        <xs:attribute ref="Name" use="required" />
    </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="CampaignLists">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="List" minOccurs="1" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

    </xs:complexType>
  </xs:element>
<!-- *** -->
<xs:element name="CampaignInfo">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="1">
      <xs:element ref="CampaignLists" />
    </xs:sequence>
    <xs:attribute ref="TenantId" use="required" />
    <xs:attribute ref="AdminLoginName" use="required" />
    <xs:attribute ref="CampaignName" use="required" />
    <xs:anyAttribute />
  </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="SyncRequest">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="CampaignInfo" />
      <xs:element ref="ListInfo" />
      <xs:element ref="RecordInfo" />
    </xs:choice>
    <xs:attribute ref="Name" use="required" />
    <xs:attribute ref="Id" use="required" />
    <xs:anyAttribute />
  </xs:complexType>
</xs:element>
<!-- *** -->
</xs:schema>

```

Campaign Synchronization Request Types

The root element for all Adapter inbound messages is `SyncRequest`. The `SyncRequest` element always contains only *one* child element with three possible types:

- `CampaignInfo`
- `ListInfo`
- `RecordInfo`

All requests can be divided into one of three types according to the *type* of this child element. Each `SyncRequest` has two mandatory attributes: `Name` and `Id`. The `Name` attribute defines the name of the Genesys request; the type of action, such as Genesys - Create Campaign. The second attribute, `Id` is intended for identification purposes.

CampaignInfo Requests and Attributes

[Table 5](#) shows the Genesys `CampaignInfo` Requests:

Table 5: Genesys CampaignInfo Requests

Name	Description
Genesys - Create Campaign	Creates all CLE objects for the requested campaign (Campaign, Calling lists, Table Accesses). Also creates OCS tables if needed.
Genesys - Delete Campaign	Deletes the Campaign CLE object
Genesys - Get Call Results for Campaign	Returns call results for a requested campaign asynchronously (as HTTP requests)
Genesys - Get Campaign Statistics	Returns call results for a requested campaign synchronously (as HTTP response)

The `CampaignInfo` element has the following attributes:

- `TenantId`; the name of a Siebel organization. Each possible name should be mapped to some tenant in Genesys CLE (see the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide* for more details.)
- `CampaignName`; the name of a campaign in Siebel and Genesys CLE

- **AdminLoginName**; the name of a Siebel Administrator. This name is used by the Adapter for mapping purposes. Each possible name should be mapped to some folder in Genesys CLE (see the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide* for more details.)

The **CampaignInfo** element may include **List** elements with the following attributes:

- **Name**; the name of a list in Siebel and Genesys if the next attribute is not included
- **CampaignName**; (optional) name of a campaign in Siebel to which the list belongs. If this attribute is included, a resulting list name in Genesys CLE will be generated by concatenation of both names: **Name (CampaignName)**.

Here are examples of **CampaignInfo** requests:

```
<SyncRequest Name="Genesys - Create Campaign" Id="1-SSSA">
  <CampaignInfo
    TenantId="Default Organization"
    CampaignName="WS77-Campaign1"
    AdminLoginName="GPADMIN">
    <CampaignLists>
      <List
        Name="WS77-List1">
      </List>
      <List
        CampaignName="WS77-Campaign5"
        Name="WS77- List2">
      </List>
      <List
        CampaignName="WS77-Campaign5"
        Name="WS77-List5">
      </List>
    </CampaignLists>
  </CampaignInfo>
</SyncRequest>

<SyncRequest Name="Genesys - Delete Campaign" Id="1-SSSB">
  <CampaignInfo
    TenantId="Default Organization"
    CampaignName="WS77-Campaign1"
    AdminLoginName="GPADMIN">
  </CampaignInfo>
</SyncRequest>
```

```

<SyncRequest
  Name="Genesys - Get Call Results for Campaign" Id="1-SSSC">
  <CampaignInfo
    TenantId="Default Organization"
    CampaignName="WS77-Campaign1"
    AdminLoginName="GPADMIN">
  </CampaignInfo>
</SyncRequest>

```

ListInfo Requests and Attributes

This type of request is most important in the synchronization procedure. Only this type of request transfers Calling Lists records from the Siebel database to the Genesys OCS database.

[Table 6](#) shows the Genesys ListInfo Requests:

Table 6: Genesys ListInfo Requests

Name	Description
Genesys - Create List	Creates CLE objects Calling Lists and Table Accesses (if they don't exist); creates OCS tables (if needed); inserts or updates list records; deletes records, which are not included in the request
Genesys - Update List	Creates CLE objects Calling Lists & Table Accesses (if they don't exist); creates OCS tables (if needed); inserts or updates list records
Genesys - Insert List Records	Inserts list records (error if record exists)
Genesys - Update List Records	Inserts or updates list records
Genesys - Delete List Records	Deletes list records
Genesys - Cancel List Records	Sends CancelRecord messages to OCS

Table 6: Genesys ListInfo Requests (Continued)

Name	Description
Genesys - Assign List	Assigns requested lists to the given Campaign CLE object (do nothing with OCS tables)
Genesys – Get List Statistics	Returns call results for a requested Calling List synchronously (as HTTP response)

Possible attributes of the ListInfo element are:

- TenantId
- AdminLoginName
- Name
- CampaignName
- FirstPage
- LastPage.

The attributes AdminLoginName and TenantId have the same meaning as in CampaignInfo requests.

The attributes Name and CampaignName are used in the same way as in the List element of CampaignInfo requests.

The attributes FirstPage and LastPage allow the Adapter on the Genesys side to determine how to combine a sequence of messages into one Calling List. If FirstPage is true, this indicates the beginning of a Calling List. If LastPage is true this indicates the end of a Calling List. If FirstPage and LastPage are both false this indicates the middle of a Calling List. The default value for these attributes is true (when they are missing).

The ListInfo element is the only place where ContactInfo elements may be included. The ContactInfo element is a key element which delivers Calling List records to OCS tables. You can add any custom field either directly to a ContactInfo element or to its child element: Record. In the first case, all records in a chain would have the same value. Otherwise, records in the same chain may have different values.

Note: If you are using the Genesys Voice Component, the attribute `crm_campaign_id` is required.

Here are examples of ListInfo requests:


```

<SyncRequest Name="Genesys - Create List" Id="1-SSSD">
  <ListInfo
    LastPage="true"
    FirstPage="true"
    TenantId="Default Organization"
    Name="WS77-List1"
    AdminLoginName="GPADMIN">
    <ListOfContactInfo>
      <ContactInfo crm_camp_con_id="2-8ULH"
        crm_campaign_id="2-7ULX">
        <Record
          contact_info="222335559"
          contact_info_type="1"
        >
        </Record>
      </ContactInfo>
      <ContactInfo crm_camp_con_id="2-86FQ"
        crm_campaign_id="2-7ULX"
        tz_dbid="(GMT-10:00) Hawaii">
        <Record
          contact_info="3334445559"
          contact_info_type="1"
        >
        </Record>
        <Record
          contact_info="4444445559"
          contact_info_type="1"
          daily_from="57600"
          daily_till="75600"
        >
        </Record>
      </ContactInfo>
    </ListOfContactInfo>
  </ListInfo>
</SyncRequest>

<SyncRequest Name="Genesys - Delete List" Id="1-SSSE">
  <ListInfo
    TenantId="Default Organization"
    CampaignName="WS77-Campaign1"
    Name="WS77- List2"
    AdminLoginName="GPADMIN">
  </ListInfo>
</SyncRequest>

```

RecordInfo Requests and Attributes

[Table 7](#) shows the Genesys RecordInfo Requests:

Table 7: Genesys RecordInfo Requests

Name	Description
Genesys - Insert DNC List Record	Sends DoNotCall messages to OCS

This request contains two elements: CustomerId and Phone. You can send DoNotCall messages as unique customer IDs or as phone numbers.

Here is an example of a RecordInfo request:

```
<SyncRequest Name="Genesys - Insert DNC List Record" Id="1-SSSF">
  <RecordInfo>
    <CustomerId>1-4IW7</CustomerId>
    <Phone></Phone>
  </RecordInfo>
</SyncRequest>
```

Campaign Synchronization Response

The Adapter generates a response for each processing request and sends it back to a client through an HTTP connection. The root element for all Adapter responses is SyncResponse. All responses have three mandatory attributes: Name, Id, and Result. Name and Id attributes have the same values as the corresponding attributes in the request.

The attribute Result may have three possible values: (Success | NotReady | Error). For all responses with Result = Success or Result = Error the corresponding requests should be removed from the client queue. For any response with Result = NotReady the corresponding request should be kept in the client queue and resent to the Adapter after timeout.

The SyncResponse may contain two types of child elements: CallResults and SynchSummary. The CallResults element is intended for backward synchronization of the OCS fields call_result and attempt. For more information about SynchSummary see “Using Synchronization Summary” on [page 61](#) in [Chapter 5](#).

Here are examples of response messages:

```
<SyncResponse Name="Genesys - Get List Statistics" Id="1-SSSX"
  Result="Success">
  <CallResults>
    <CallResultRecord>
      <crm_camp_con_id>1-8ZGR</crm_camp_con_id>
      <call_result>0</call_result>
```

```

        <call_result_name>Ok</call_result_name>
        <attempt>4</attempt>
        <call_time>18000334555</call_time>
    </CallResultRecord>
    <CallResultRecord>
        <crm_camp_con_id>1-8ZGS</crm_camp_con_id>
        <call_result>7</call_result>
        <call_result_name>No Answer</call_result_name>
        <attempt>2</attempt>
        <call_time></call_time>
    </CallResultRecord>
</CallResults>
</SyncResponse>

<SyncResponse Name="Genesys - Create List" Id="1-SSSY" Result="Error">
    <SynchSummary>
        <SynchSummaryRecord>
            <Object>Calling List Record</Object>
            <ObjectN>11</ObjectN>
            <Operation>UPDATE</Operation>
            <OperationStatus>SUCCESS</OperationStatus>
            <AuxValue1>CLT_3842</AuxValue1>
            <AuxValue2></AuxValue2>
            <AuxValue3></AuxValue3>
        </SynchSummaryRecord>
        <SynchSummaryRecord>
            <Object>Calling List Record</Object>
            <ObjectN>1</ObjectN>
            <Operation>DELETE</Operation>
            <OperationStatus>FAILURE</OperationStatus>
            <AuxValue1>CLT_3842</AuxValue1>
            <AuxValue2></AuxValue2>
            <AuxValue3></AuxValue3>
        </SynchSummaryRecord>
    </SynchSummary>
</SyncResponse>

```

The formal definition of the Adapter's responses, the Outbound XML Schema, is shown in the next section.

The Outbound XML Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Gplus Adapter 7.5. Campaign Synchronization Schema of Outbound Messages-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definition of attributes -->
  <xs:attribute name="Name" type="xs:string" />
  <xs:attribute name="Id" type="xs:string" />
  <xs:attribute name="Result">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Success"/>
        <xs:enumeration value="Error"/>
        <xs:enumeration value="NotReady"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="ErrorCode" type="xs:string" />
  <xs:attribute name="ErrorText" type="xs:string" />
  <!-- *** -->
  <!-- definition of elements -->
  <xs:element name="crm_camp_con_id" type="xs:string" />
  <xs:element name="call_result" type="xs:integer" />
  <xs:element name="call_result_name" type="xs:string" />
  <xs:element name="call_time" type="xs:integer" />
  <xs:element name="attempt" type="xs:integer" />
  <xs:element name="CallResultRecord">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="crm_camp_con_id" />
        <xs:element ref="call_result" />
        <xs:element ref="call_result_name" />
        <xs:element ref="call_time" />
        <xs:element ref="attempt" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!-- *** -->
  <xs:element name="Object" type="xs:string" />
  <xs:element name="ObjectN" type="xs:integer" />
  <xs:element name="Operation" type="xs:string" />
  <xs:element name="OperationStatus" type="xs:string" />
  <xs:element name="AuxValue1" type="xs:string" />
  <xs:element name="AuxValue2" type="xs:string" />
  <xs:element name="AuxValue3" type="xs:string" />

```

```

<xs:element name="SynchSummaryRecord">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Object" />
      <xs:element ref="ObjectN" />
      <xs:element ref="Operation" />
      <xs:element ref="OperationStatus" />
      <xs:element ref="AuxValue1" />
      <xs:element ref="AuxValue2" />
      <xs:element ref="AuxValue3" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="CallResults">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="CallResultRecord" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="SyncSummary">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="SynchSummaryRecord" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- *** -->
<xs:element name="SyncResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="CallResults" minOccurs="0" maxOccurs="1" />
      <xs:element ref="SyncSummary" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute ref="Name" use="required" />
    <xs:attribute ref="Id" use="required" />
    <xs:attribute ref="Result" use="required" />
    <xs:attribute ref="ErrorCode" use="optional" />
    <xs:attribute ref="ErrorText" use="optional" />
    <xs:anyAttribute />
  </xs:complexType>
</xs:element>
<!-- *** -->
</xs:schema>

```




Chapter

3

Configuring the List Import Functionality

This chapter provides guidelines for configuring the Campaign Synchronization Component's list import functionality. It contains the following sections:

- [Siebel Configuration Guidelines, page 39](#)
- [Format Configuration Guidelines, page 40](#)
- [Style Sheet Configuration Guidelines, page 43](#)
- [Style Sheet Configuration Examples, page 45](#)

Siebel Configuration Guidelines

You should make sure that all the Siebel fields that you plan to export to the Genesys environment are passed from the Siebel environment to the Campaign Synchronization Component. To verify this, use the Campaign Synchronization Component's log to examine the Siebel XML received by the Campaign Synchronization Component. (Siebel XML is printed in the log when the verbose application option is set to debug). All the required fields must be attributes of the `ContactInfo` or the `Record` XML elements.

If this is not the case, use Siebel Tools to perform the following actions:

1. Check that the Campaign List Contact Business Component contains the required fields and that those fields are active. If necessary, modify the Business Component to include or activate the fields.
2. For the Genesys - Campaign List Contact Integration Object, check that the `ContactInfo` Integration Component (external name - Campaign List Contact) contains the required fields, and check that those fields are active. Check also that the "Name" column of the "Integration Component Fields" table has appropriate values as shown in Table 8 on [page 40](#).

Table 8: Default Mapping of Siebel and Genesys Names in the Campaign List Contact Integration Component

External Name (Siebel)	Name (Genesys)
Calculated Home Phone #	phone_1
Calculated Work Phone #	phone_2
CampaignId	crm_campaign_id
Id	crm_camp_con_id
Calculated Contact Id	crm_contact_id
Calculated Time Zone Name	tz_dbid

If some of the fields are absent, synchronize the Integration Object using the Synchronize button in the Integration Objects window. Then, after synchronization, change the type for the Id Integration Component Field from System to Data. If some of the fields are not active, activate them. Correct values of the “Name” column are shown in [Table 8](#).

3. If you make any changes to the Siebel Repository as a result of steps 1 and 2 above, update the Siebel Repository File by compiling the Campaign Synchronization project, and deploy the updated file on the Siebel Server.

The steps listed above are general guidelines only. For detailed information on how to use Siebel Tools to modify Business Components, Integration Components, Integration Objects, and the Siebel Repository File, please see your Siebel documentation.

Format Configuration Guidelines

When the Campaign Synchronization Component imports Siebel Lists to the Genesys environment, the format object that it uses is specified by: Configuration Manager > Campaign Synchronization Component application object > options > Genesys section > format option. If you work in a multi-tenant Genesys environment, you should create a copy of the format object under each Genesys Tenant that is mapped to a Siebel Organization or Division.

The list import format must include the fields listed in [Table 9](#). These fields are essential for the Adapter and Genesys Outbound Contact to function properly.

Note: When you use `<datetime>` types of fields in your synchronization process, you *must* check that input formats of these fields are equal for both the Siebel and Genesys databases. If this is not the case, some additional transformation is required; values of these fields in XML messages should meet input format requirements for the OCS database. This transformation can be accomplished by using an XSL tool (a style sheet file) on Genesys side (see “Style Sheet Configuration Guidelines” on [page 43](#) for more details).

Table 9: Genesys Fields That Must Be Included in the Format Object

Genesys OCS Field
record_id
contact_info
contact_info_type
record_type
record_status
call_result
attempt
dial_sched_time
call_time
daily_from
daily_till
tz_dbid
campaign_id
agent_id
chain_id
chain_n
crm_campaign_id
crm_camp_con_id
crm_contact_id

Table 9: Genesys Fields That Must Be Included in the Format Object (Continued)

Genesys OCS Field
app_id
group_id
media_ref
treatments

If you want to add some additional contact fields to the synchronization process, you have to add them in both Siebel and Genesys environments:

1. Create new format field objects in the Genesys environment. See the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide*.
2. Activate the fields in the ContactInfo Integration Component.
3. Rename those fields in the ContactInfo Integration Component according to their names in Genesys.
4. Compile the Genesys Campaign Synchronization project and deploy the updated `srf-file` on the Siebel Server.

If you want to synchronize additional contact info fields such as “Email Address” or “Fax Phone #,” you have to name them as `phone_X` where `X` is the proper value for the Genesys contact info type. Refer to [Table 10](#) for these Contact Info values. The highest value should not exceed the value of the last type (10).

Table 10: Genesys Contact Info Types

Name	Value
HomePhone	1
DirectBusinessPhone	2
BusinessWithExt	3
Mobile	4
VacationPhone	5
Pager	6

Table 10: Genesys Contact Info Types (Continued)

Name	Value
Modem	7
VoiceMail	8
PinPage	9
E-mail	10

Style Sheet Configuration Guidelines

In case you need to transform some values of synchronized data such as phone # by your own rules, or add calculated fields for each record in the chain, you can use the `xmlTransformer` option of the Campaign Synchronization Component. In this case you have to add this option to the Genesys section, see the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide*. The Component provides the mechanism to customize the transformation rules through a style sheet definition.

When you create a style sheet that the Campaign Synchronization Component will use to import Siebel Lists, the style sheet must map a set of predefined Genesys fields, which are listed in [Table 11](#).

Table 11: Genesys Fields that the Style Sheet Must Map

Genesys Field
contact_info
contact_info_type
crm_campaign_id
crm_camp_con_id
crm_contact_id
tz_dbid

The Campaign Synchronization Component package contains the `GplusCampSynch.xml`, which you can use as a template to create your own style sheet. This style sheet is an example, which replaces any values of the `TenantId` field (Siebel Organization) to the constant `Default Organization`. This example may be useful for customers who would like to manage all Siebel organizations as one Genesys tenant in Genesys CLE.

The file content is shown below:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="windows-1252"/>
  <xsl:template match="/ | @* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>
  <xsl:template match="@TenantId">
    <xsl:attribute name="TenantId">Default Organization</xsl:attribute>
  </xsl:template>
</xsl:stylesheet>
```

The Campaign Synchronization Component automatically generates values for some of the predefined Genesys fields. As a result, the style sheet must *not* map these fields, which are listed in [Table 12](#).

Table 12: Genesys Fields That You Cannot Map Using the Style Sheet

Genesys Field
record_id
chain_id

When requested to update Genesys Calling List records, the Campaign Synchronization Component updates all Genesys fields mapped by the style sheet, with the exception of the fields shown in [Table 13](#).

Table 13: Genesys Fields That You Cannot Update Using the Style Sheet

Genesys Field
record_id
chain_id
crm_campaign_id
crm_camp_con_id

You must also ensure that the corresponding format object includes a field definition for each Genesys field that the style sheet maps.

Style Sheet Configuration Examples

Example 1

The code listing in this section illustrates how to configure the list import functionality to replace a prefix “XXX”, in phone numbers imported from Siebel, with an access code “123” in phone numbers stored in Genesys Calling Lists. If you use dialing filters in your Siebel environment, you will need to perform similar processing on imported phone numbers.

The corresponding style sheet file is TransformXXX.xsl. It was created by copying the GplusCampSynch.xsl file (which is a part of the Campaign Synchronization Component installation package) and adding the elements shown in **bold** in order to perform the substitution.

Warning! The following code listing simply illustrates a style sheet file with entries to replace one substring with another. Genesys does not support this modified style sheet. If you need detailed instructions and support for customizing a style sheet, please contact Genesys Professional Services.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="windows-1252"/>
  <xsl:template match="/ | @* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>
  <xsl:template match="*contact_info">
    <xsl:attribute name="contact_info">
      <xsl:choose>
        <xsl:when test="starts-with(., 'XXX')">
          <xsl:value-of select="concat('123', substring-after(.,
            'XXX'))"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="."/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
  </xsl:template>
  <xsl:template match="Phone">
    <Phone>
      <xsl:choose>
        <xsl:when test="starts-with(., 'XXX')">
          <xsl:value-of select="concat('123', substring-after(.,
            'XXX'))"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="."/>
        </xsl:otherwise>
      </xsl:choose>
    </Phone>
  </xsl:template>
</xsl:stylesheet>
```

```

        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="."/>
        </xsl:otherwise>
    </xsl:choose>
</Phone>
</xsl:template>

</xsl:stylesheet>

```

Example 2

The code listing in this section illustrates how to configure the list import functionality to generate field values for each OCS Record based on `contact_info_type`.

Suppose that a customer has two additional requirements:

- Change the order of records in a chain based on phone type; work phone should be first and home phone should be second.
- Set values of `daily_from` and `daily_till` according to phone type; for work phones `daily_from = 8am` and `daily_till = 6pm`; for home phones `daily_from = 4pm` and `daily_till = 9pm`.

The first requirement can be done by inserting the `chain_n` attribute, with value calculated according to `contact_info_type`.

The corresponding style sheet file is `TransformRecordAttr.xsl`. It was created by copying the `GplusCampSynch.xsl` file (which is a part of the Campaign Synchronization Component installation package) and adding the elements shown in bold.

Warning! The following code listing simply illustrates a style sheet file with entries to generate field values for each OCS record. Genesys does *not* support this modified style sheet. If you need detailed instructions and support for customizing a style sheet, please contact Genesys Professional Services.

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output encoding="windows-1252"/>
    <xsl:template match="/ | @* | node()">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>
    <xsl:template match="@contact_info_type">
        <xsl:attribute name="contact_info_type"><xsl:value-of select="."/>
    </xsl:attribute>
        <xsl:attribute name="chain_n">

```

```

    <xsl:choose>
      <xsl:when test=". = '1'">
        <xsl:value-of select="1"/>
      </xsl:when>
      <xsl:when test=". = '2'">
        <xsl:value-of select="0"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="2"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
  <xsl:attribute name="daily_from">
    <xsl:choose>
      <xsl:when test=". = '2'">
        <xsl:value-of select="28800"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="57600"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
  <xsl:attribute name="daily_till">
    <xsl:choose>
      <xsl:when test=". = '2'">
        <xsl:value-of select="64800"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="75600"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
</xsl:template>
</xsl:stylesheet>

```




Chapter

4

Using the Genesys Campaign Synchronization Business Service

This chapter provides general guidelines on how to use the Genesys Campaign Synchronization Business Service to develop campaign synchronization scenarios between the Siebel and Genesys applications. It also describes the methods that this Business Service exposes, and presents an example of Siebel eScript that invokes some of those methods. The chapter contains the following sections:

- [Method Usage Guidelines, page 50](#)
- [Method Descriptions, page 50](#)
- [Script Example, page 56](#)

Note: Default scenarios of the Genesys *Gplus* Campaign Synchronization component, based on Runtime events, implement real-time synchronization between Siebel and Genesys databases. This means that all changes in assigned Siebel Calling Lists automatically transfer to the Genesys environment. When you use your own scenarios, you must disable this automatic update generated by Genesys Runtime Events to avoid double synchronization. In this case, you should *not* install Genesys Runtime events in order to prevent automatic update by default scenarios.

Method Usage Guidelines

Most of the Genesys Campaign Synchronization Business Service methods take either one or two string arguments. The arguments mentioned in the following descriptions are required except where listed as optional.

A call to the `PreSubmitRequests` method must precede calls to any other methods of the Genesys Campaign Synchronization Business Service, and a call to the `SubmitRequests` method must follow all other Genesys Campaign Synchronization Business Service method calls. You can think of this required calling order as analogous to a transaction, in which `PreSubmitRequests` defines the beginning of a group of related operations, and `SubmitRequests` commits the whole group. However, this is just an analogy—these two methods do not actually implement transactional behavior.

Method Descriptions

The entries in this section describe the following methods of the Genesys Campaign Synchronization Business Service:

- [PreSubmitRequests, page 50](#)
- [SubmitRequests, page 51](#)
- [ExportCampaign, page 51](#)
- [DeleteCampaign, page 51](#)
- [ExportLists, page 52](#)
- [AppendList, page 52](#)
- [DeleteLists, page 53](#)
- [AssignLists, page 53](#)
- [ExportCampaignWithLists, page 53](#)
- [DeleteCampaignWithLists, page 54](#)
- [InsertListRecords, page 54](#)
- [UpdateListRecords, page 55](#)
- [DeleteListRecords, page 55](#)
- [InsertDNCListRecord, page 55](#)
- [DeleteDNCListRecord \(obsolete\), page 56](#)

PreSubmitRequests

A call to this method should precede any other calls to methods of the Genesys Campaign Synchronization Business Service.

Argument

[None.]

SubmitRequests

A call to this method should follow any other calls to methods of the Genesys Campaign Synchronization Business Service. When this method is invoked, all requests made after the last `PreSubmitRequests` method call are sent to the Campaign Synchronization Component. For related information, see “Using Synchronization Summary” on [page 61](#).

Arguments

<code>ReferenceValue1</code>	Optional: User-defined value that you can use to select records from the Genesys Synch Summary Business Component after synchronization is complete.
<code>ReferenceValue2</code>	Optional: User-defined value that you can use to select records from the Genesys Synch Summary Business Component after synchronization is complete.
<code>ReferenceValue3</code>	Optional: User-defined value that you can use to select records from the Genesys Synch Summary Business Component after synchronization is complete.

ExportCampaign

This method creates a request to the Genesys Campaign Synchronization component to export a specified Siebel Campaign to the Genesys environment. For the exported Campaign, the Campaign Synchronization component creates a Campaign object in the Genesys Configuration Layer. This object is placed in a Campaign folder assigned to the Siebel user who created this campaign. Also, this method adds to the campaign object a list of all Siebel segments/lists assigned to this campaign.

Argument

<code>CampaignId</code>	A string representing the ID of the Siebel Campaign that will be exported to the Genesys environment.
-------------------------	---

DeleteCampaign

This method creates a request to the Campaign Synchronization component to delete a specified, exported Siebel Campaign from the Genesys environment.

The Campaign Synchronization component deletes the corresponding Campaign object from the Genesys Configuration Layer.

Argument

CampaignId A string representing the ID of the Siebel Campaign that will be deleted from the Genesys environment.

ExportLists

This method creates a request to the Campaign Synchronization component to export to the Genesys environment all Lists that are assigned to a specified Siebel Campaign. Contacts directly assigned to the Siebel Campaign are exported as a separate List. For each exported List, the Campaign Synchronization component:

- Creates a Table Access object in Genesys Configuration Layer. This object is placed in a Table Access folder assigned to the Siebel user performing the export.
- Creates a Calling List object in Genesys Configuration Layer. This object is created in a Calling List folder assigned to the Siebel user performing the export.
- Creates a database table, and a set of indexes related to that table, located in a database corresponding to the Database Access Point assignment of the Siebel user performing the export.
- Populates the created database table with records based on Contact information from the List.

Argument

CampaignId A string representing the ID of the Siebel Campaign whose Lists will be exported to the Genesys environment.

AppendList

This method creates a request to the Campaign Synchronization Component to append an existing List, in the Genesys environment, that corresponds to the Siebel List specified by the **CampaignId** and **ListId** parameters. The only records appended are those based on Contacts that are absent from the Genesys List.

Arguments

CampaignId A string representing the ID of the Siebel Campaign to which the Siebel List belongs.

ListId (Optional argument:) A string representing the ID of the Siebel List. If this argument is not specified, the Campaign Synchronization Component appends Contacts that are directly assigned to the specified Siebel Campaign, but absent from the corresponding Genesys List.

DeleteLists

This method creates a request to the Campaign Synchronization component to delete from the Genesys environment all exported Lists that are assigned to the specified Siebel Campaign. This also includes the List for Contacts directly assigned to the Siebel Campaign. For each deleted List, the Campaign Synchronization component:

- Deletes the corresponding Calling List object from the Genesys Configuration Layer.
- Deletes the corresponding Table Access object from the Genesys Configuration Layer.
- Drops the related database table with the List records.

Argument

CampaignId A string representing the ID of the Siebel Campaign whose Lists will be deleted from the Genesys environment.

AssignLists

This method is obsolete. All Siebel segments/lists are assigned to a campaign object in the Genesys environment by the `ExportCampaign` method.

Arguments

CampaignId A string representing the ID of the Siebel Campaign whose exported Lists will be assigned to another exported Siebel Campaign in the Genesys environment.

ToCampaignId A string representing the ID of the second exported Siebel Campaign. This is the Campaign to which the Lists should be reassigned in the Genesys environment.

ExportCampaignWithLists

This method creates a request to the Campaign Synchronization component to export a given Siebel Campaign, and all Lists assigned to this Campaign, to the

Genesys environment. The exported Lists are then assigned to the corresponding exported Campaign in the Genesys environment.

In its implementation, this method uses the following related methods. For additional details, please see the descriptions for those methods:

- [ExportCampaign \(see page 51\)](#)
- [ExportLists \(see page 52\)](#)

Argument

CampaignId	A string representing the ID of the Siebel Campaign whose definition and Lists will be exported to the Genesys environment.
------------	---

DeleteCampaignWithLists

This method creates a request to the Campaign Synchronization component to delete from the Genesys environment a specified, exported Siebel Campaign and all exported Lists assigned to this Campaign. In its implementation, this method uses the following related methods. For additional details, please see the descriptions for those methods:

- [DeleteCampaign \(see page 51\)](#)
- [DeleteLists \(see page 53\)](#)

Argument

CampaignId	A string representing the ID of the Siebel Campaign whose definition and Lists will be deleted from the Genesys environment.
------------	--

InsertListRecords

This method creates a request to the Campaign Synchronization component to insert records for a specified Siebel Campaign List Contact into the corresponding List in the Genesys environment. The List to be updated is determined based on the Siebel Campaign, and on the Siebel List, to which the Campaign List Contact is assigned.

Argument

CampaignListContactId	A string representing the ID of the Siebel Campaign List Contact whose records will be inserted in the Genesys environment.
-----------------------	---

UpdateListRecords

This method creates a request to the Campaign Synchronization component to update records in the Genesys environment for a specified Siebel Campaign List Contact. The Genesys List to be updated is determined based on the Siebel Campaign, and on the Siebel List, to which the Campaign List Contact is assigned.

Argument

<code>CampaignListContactId</code>	A string representing the ID of the Siebel Campaign List Contact whose records will be updated in the Genesys environment.
------------------------------------	--

DeleteListRecords

The `DeleteListRecords` method creates a request to the Campaign Synchronization component to delete records from the Genesys environment for a specified Siebel Campaign List Contact. The Genesys List to be updated is determined based on the Siebel Campaign, and on the Siebel List, to which the Campaign List Contact is assigned.

Argument

<code>CampaignListContactId</code>	A string representing the ID of the Siebel Campaign List Contact whose records will be deleted in the Genesys environment.
------------------------------------	--

InsertDNCListRecord

This method creates a request to the Campaign Synchronization component to insert a given customer ID or phone number into the “Do Not Call” list in the Genesys environment.

Argument

<code>CustomerId</code>	(Optional argument:) A string representing the customer ID that will be inserted into the “Do Not Call” list in the Genesys environment.
<code>Phone</code>	(Optional argument:) A string representing the phone number that will be inserted into the “Do Not Call” list in the Genesys environment.

DeleteDNCListRecord (obsolete)

This method is obsolete. The method is left in the API for backward compatibility with older versions of the Campaign Synchronization (Outbound) component.

Argument

CustomerId	(Optional argument:) A string representing the customer ID that will be deleted from the “Do Not Call” list in the Genesys environment.
Phone	(Optional argument:) A string representing the phone number that will be deleted from the “Do Not Call” list in the Genesys environment.

Script Example

The following eScript listing provides an example of campaign synchronization from the Siebel to the Genesys environment. The script invokes the following methods of the Genesys Campaign Synchronization Business Service:

- PreSubmitRequests
- ExportCampaignWithLists
- ExportLists
- SubmitRequests

Comments embedded in the script clarify the flow of control.

Note: This script is provided as an illustration only, and is not supported by Genesys.

```
function ExportCampaignH(CampaignId)
{
    // Function to export a given Siebel Campaign and its immediate
    children
    // to Genesys environment using Genesys Campaign Synchronization
    Business Service in
    // Compatibility mode.
    //
    // Function does not create Genesys Campaign objects for the
    Child
    // Campaigns of the exported Siebel Campaign. All Lists
    // from the Child Campaigns in Siebel are assigned to the
```



```

// exported Siebel Campaign in Genesys.

var svGenesys = TheApplication().GetService("Genesys Campaign
Synchronization");

var psInput = TheApplication().NewPropertySet();
var psOutput = TheApplication().NewPropertySet();
svGenesys.InvokeMethod("PreSubmitRequests", psInput, psOutput);

// Create a request to export the specified Siebel Campaign with
its
// Lists to Genesys
// NOTE: In Compatibility mode, all lists of the exported
Campaign
// and its Child campaigns have been assigned to the Campaign
here.
// In Regular mode, the Genesys Adapter assigns lists of the
current Campaign only.

psInput.SetProperty("CampaignId", CampaignId);
svGenesys.InvokeMethod("ExportCampaignWithLists", psInput,
psOutput);

// Determine all children of the exported Siebel Campaign

var boCampaign = TheApplication().GetBusObject("Campaign");
var bcCampaign = boCampaign.GetBusComp("Campaign");
var bIsRecord;

with (bcCampaign)
{
    ActivateField("Parent Source Id");
    ClearToQuery();
    SetSearchSpec("Parent Source Id", CampaignId);
    ExecuteQuery(ForwardOnly);
    bIsRecord = FirstRecord();
}

var sChildCampaignId;

while (bIsRecord)
{
    // For each Child Campaign, create a request to export its
Lists
    // to Genesys

    sChildCampaignId = bcCampaign.GetFieldValue("Id");

    psInput = TheApplication().NewPropertySet();
    psOutput = TheApplication().NewPropertySet();
    psInput.SetProperty("CampaignId", sChildCampaignId);

```

```
        svGenesys.InvokeMethod("ExportLists", psInput, psOutput);

        bIsRecord = bcCampaign.NextRecord();
    }

    // Submit all created requests for execution

    psInput = TheApplication().NewPropertySet();
    psOutput = TheApplication().NewPropertySet();
    with (psInput)
    {
        SetProperty("ReferenceValue1", "ExportCampaignWithLists");
        SetProperty("ReferenceValue2", CampaignId);
        //SetProperty("ReferenceValue3", "Something");
    }
    svGenesys.InvokeMethod("SubmitRequests", psInput, psOutput);

    sChildCampaignId = null;
    bIsRecord = null;
    bcCampaign = null;
    boCampaign = null;
    psOutput = null;
    psInput = null;
    svGenesys = null;
}
```



Chapter

5

Synchronization Summary Usage

Summary information about results of data synchronization requests is collected in the Siebel environment by using the Genesys Synch Summary Siebel Business Component. This chapter provides guidelines for using the Genesys Synchronization Summary functionality. It contains the following sections:

- [Synchronization Summary Format, page 59](#)
- [Using Synchronization Summary, page 61](#)

Synchronization Summary Format

The Campaign Synchronization application (running in the Genesys environment) sends synchronization summary information back to the Siebel environment for each data synchronization request in the body of the corresponding response. Synchronization summary is an XML document with the structure described by the following table.

Table 14: Structure of Synchronization Summary XML Document

Field	Description
Object	Type of object in Genesys affected by the data synchronization request (Campaign, Calling List, Calling List Record, etc.)
ObjectN	Number of Genesys objects of the same type affected by the data synchronization request
Operations	Performed operation (ADD, CHANGE, DELETE, etc.)
OperationsStatus	Status of the performed operation (SUCCESS, FAILURE)
AuxValue1	Additional information that varies based on the request
AuxValue2	Additional information that varies based on the request
AuxValue3	Additional information that varies based on the request

The Genesys Synch Summary Business Component, which is used to collect synchronization summary information, has fields similar to those in the table above. In addition, this Business Component has three “Reference Value” fields to simplify record selection. The mapping between Genesys Synch Summary Business Component fields and columns from its base table, CX_GEN_SYN_SUM, is shown below.

Table 15: Synchronization Summary Business Component Map

Genesys Synch Summary Business Component Field	CX_GEN_SYN_SUM Table Field	CX_GEN_SYN_SUM Table Field Type
Object	OBJECT	varchar(30)
ObjectN	OBJECT_N	numeric(22,10)
Operations	OPERATION	varchar(30)
OperationsStatus	OPERATION_ST	varchar(30)
AuxValue1	AUX_VAL1	varchar(75)
AuxValue2	AUX_VAL2	varchar(75)
AuxValue3	AUX_VAL3	varchar(75)
Reference Value 1	REF_VAL1	varchar(75)
Reference Value 2	REF_VAL2	varchar(75)
Reference Value 3	REF_VAL3	varchar(75)

Using Synchronization Summary

To use synchronization summary, you work with the `SubmitRequests` method of the Genesys Campaign Synchronization Siebel Business Service. This method provides the following optional input arguments: `ReferenceValue1`, `ReferenceValue2`, and `ReferenceValue3`. These contain user-defined values that you can use to select records from the Genesys Synch Summary Business Component after synchronization is complete. Refer to [“SubmitRequests” on page 51](#).

The following Siebel eScript code fragment illustrates how these arguments can be used:

```
function ExportList (CampaignId, ListId)
{
    // Obtain "Genesys Campaign Synchronization" Business Service
    var svGenesys =
        TheApplication().GetService("Genesys Campaign Synchronization");

    //
    // Prepare the service for request submission
```

```

//
var psInput = TheApplication().NewPropertySet();
var psOutput = TheApplication().NewPropertySet();
svGenesys.InvokeMethod("PreSubmitRequests", psInput, psOutput);

//
// Create a request to export the specified
// Siebel Campaign List
//
psInput = TheApplication().NewPropertySet();
psOutput = TheApplication().NewPropertySet();
psInput.SetProperty("CampaignId", CampaignId);
psInput.SetProperty("ListId", ListId);
svGenesys.InvokeMethod("ExportList", psInput, psOutput);

//
// Submit the request for execution
//
psInput = TheApplication().NewPropertySet();
psOutput = TheApplication().NewPropertySet();
psInput.SetProperty("ReferenceValue1", "LIST EXPORT");
svGenesys.InvokeMethod("SubmitRequests", psInput, psOutput);

psOutput = null;
psInput = null;
svGenesys = null;
}

```

Data synchronization requests submitted to the Campaign Synchronization Component using the `PreSubmitRequests` and `SubmitRequests` methods of the Genesys Campaign Synchronization Business Service will complete asynchronously.

Upon completion of each request issued between `PreSubmitRequests` and `SubmitRequests`, the implementation calls the `Appended` method of the Genesys Campaign Synchronization Business Service. The call to this method provides a call back mechanism that allows you to implement customized processing (such as showing a Siebel view with synchronization results, sending an e-mail message, and so on) of data synchronization results stored in the Genesys Synch Summary Business Component. You can do this by modifying the default implementation of the `Appended` method.

You can select relevant synchronization summary information by specifying values for the “Reference Value” arguments. You can enter a value as any string that does not exceed 75 characters in length. The `SubmitRequests` method passes these values to the `Appended` method.

Following is a hypothetical implementation of the `Appended` method of the Genesys Campaign Synchronization Business Service that selects

synchronization summary records that correspond to the data synchronization requests issued by the `ExportList` function above:

```
function Appended(RefValue1, RefValue2, RefValue3)
{
    //
    // Obtain Genesys Synch Summary Business Component
    //
    var boGenSynchSummary
    = TheApplication().GetBusObject("Genesys Synch Summary");
    var bcGenSynchSummary
    = boGenSynchSummary.GetBusComp("Genesys Synch Summary");

    //
    // Query Genesys Synch Summary
    //
    with (bcGenSynchSummary)
    {
        ActivateField("Reference Value 1");
        ActivateField("Object");
        ActivateField("Object #");
        ActivateField("Operation");
        ActivateField("Operation Status");
        ClearToQuery();
        SetSearchSpec("Reference Value 1", RefValue1);
        ExecuteQuery(ForwardOnly);
    }

    //
    // Do something with selected records
    //
    ...

    bcGenSynchSummary = null;
    boGenSynchSummary = null;
}
```




Chapter

6

Media Routing Component Customization

The Media Routing Component integrates the Siebel and Genesys software's handling of e-mail interactions.

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

- [Overview, page 65](#)
- [Using the GplusMediaRoute Business Service, page 66](#)
- [Applet Customization, page 74](#)
- [Using the Media Routing Component for Routing Siebel Work Items, page 74](#)

Overview

The Media Routing Component version 7.5 supports a relation between Siebel activities and Genesys interactions, by means of a one-to-one relationship between `InteractionId` and `ThirdPartyId`, where `InteractionId` is for a Genesys interaction ID, and `ThirdPartyId` is for a Siebel eMail activity record ID. The `Call Id` field of a Siebel eMail activity is used to store a Genesys `InteractionId`, and the Genesys interaction contains the `ThirdPartyId` in attached data.

The relationship between Siebel activities and Genesys interactions is essential for Pull/Stop functionality. However, if you do not use this functionality, you may choose not to support this relationship in your customization (for example, you may choose not to store a Genesys `InteractionId` in a Siebel eMail activity record). A Genesys interaction must have `ThirdPartyId` in any case. Please see the description of the “Route Method” on [page 71](#). For your customization you may use any unique Siebel record field value as the `ThirdPartyId`.

When a `GplusMediaRouting-ProcessMessage` workflow sends a route request, it writes `InteractionId` in a `Call Id` field and changes the status of the Siebel activity. If it is a successful route request, activity status will be `Queued`, otherwise it will be `NotQueued`.

The `Call Id` field is used in `Pull` and `Stop Interaction` commands to get the `InteractionId` for the command. `ThirdPartyId` along with `MediaType` are used to open a proper view and to locate a proper Siebel record when an agent accepts an incoming interaction.

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 `InteractionId`, you must add a new field into the business component using custom extension columns or an extension table. For more information, please refer to the *Siebel Tools Reference* from Siebel.

Using the GplusMediaRoute Business Service

The `GplusMediaRoute Business Service` is a business service which performs routing of Siebel work items and updating of Siebel activity records. The `GplusMediaRoute Business Service` contains the following methods:

- `GetTopWorkItem`
- `MarkWorkItemDone`
- `PullInteraction`
- `StopWorkItem`
- `route`
- `UpdateActivity`

To enable a debug log, you may add a `DebugLogFile` input parameter and set it to a debug log file name. This business service should not be used for email or chat media types.

GetTopWorkItem Method

The `GetTopWorkItem` method is used to get information about a top active work item. It does not have input parameters. Output parameters are `InteractionId`, `MediaType`, and `ThirdPartyId`. For the meaning of these parameters please see the “Overview” on [page 65](#).

MarkWorkItemDone and Special MarkDone Methods

The `MarkWorkItemDone` method is used to mark a Siebel work item as done. The method checks what active workitem is present, and if the `MediaType` input parameter is set, it checks that the active workitem has the proper media type.

If the `QueueName` input parameter is set it will be used, otherwise `QueueParameterName` will be used. The method sets a `MarkDoneQueue` output parameter and invokes the `MarkDoneMR` command from the communication configuration.

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

For convenience, several `MarkDone` methods are provided by the `GplusMediaRoute` business service. All `MarkDone` methods perform the same actions, have the same parameter list, and are enabled only if an active workitem is present and its media type is neither email nor chat. However, note the following:

- The `MarkWorkSE` method is enabled if the active workitem media type is `SiebelEmail` or `BackgroundEmail`.
- The `MarkWorkSO` method is enabled if the active workitem media type is `ServiceOrder`.
- The `MarkWorkSR` method is enabled if the active workitem media type is `ServiceRequest`.
- The `MarkWorkItemDone` method is enabled if the active workitem media type is neither `SiebelEmail` nor `BackgroundEmail`.
- There is also a `MarkDone custom-media-type` method, where `custom-media-type` is any custom media type. The value of `custom-media-type` should not contain any spaces, but there *should* be a space between `MarkDone` and `custom-media-type`. The `MarkDone custom-media-type` method is enabled if the active workitem media type is `custom-media-type`.

For a sample that uses these methods, please refer to the `MarkDone-SO` and `MarkDone-SR` commands in the file `OBJECTS/GenComm_universal.def`.

For a list of input parameters and their descriptions, see Table 16 on [page 68](#).

Table 16: MarkDone Methods: Input Parameters

Parameter	Required	Default Value	Description
MediaType	No		The list of media types, separated by commas. If this parameter is set, only interactions of media types specified in the list will be placed in the queue.
RecIdField	No		The field name which stores the <code>ThirdPartyId</code> . If it is set, the method checks whether this field value is equal to the <code>ThirdPartyId</code> of the work item. If it is not equal, the command will be cancelled. This parameter may be used to prevent the method from being invoked for a wrong work item, such as to ensure that a selected work item corresponds to a selected record, if it is required.
QueueName	No	<code>__STOP__</code>	The name of the queue where the interaction should be placed. <code>__STOP__</code> is a special name, which means that the interaction processing by Genesys will be stopped.
QueueParameterName	No	<code>MediaRoutingDoneQueue</code>	This is an alternative way to set a queue name. This value is the name of a communication configuration parameter whose value will be used as a queue name.
Status	No	<code>-</code>	Record status to update. If it is set, the method will do an update of the record status.

Table 16: MarkDone Methods: Input Parameters (Continued)

Parameter	Required	Default Value	Description
StatusField	No	Status	Field name used to store a record status
FilterMediaType	No		List of media types, separated by commas. If this parameter is set, the method checks what interaction media type belongs to this list. If not, the command will be cancelled.

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 Router.

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

For a list of input parameters and their descriptions, see [Table 17](#).

Table 17: PullInteraction Method Input Parameters

Parameter	Required	Default Value	Description
InteractionId	Yes, if <code>ThirdPartyId</code> is absent		Genesys interaction ID to be used to pull the interaction
ThirdPartyId	Yes, if <code>InteractionId</code> is absent		ID to be used as the Siebel record ID. See the “ <code>RecIdField</code> ” on page 70 .
InteractionField	No	Call Id	Field name used to store Genesys interaction ID

Table 17: PullInteraction Method Input Parameters (Continued)

Parameter	Required	Default Value	Description
RecIdField	No	Id	Field name to be used as ThirdPartyId
BusObject	No	Action	Business object name
BusComp	No	Action	Business component name

StopWorkItem Method

The StopWorkItem method is used to cancel a route request, stopping it from being processed in the Genesys environment. The method also may update the status of a Siebel record and assign it to the agent. Refer to Table 18 on [page 70](#) for details.

Note: The method first assigns the record to the agent if the UserName parameter is set; when to stop processing and when to change the record status if the SuccessStatus or FailedStatus parameter is set. The interaction can not be pulled if the interaction is being processed by another agent or by Genesys Router.

Table 18: StopWorkItem Method Input Parameters

Parameter	Required	Default Value	Description
PrimaryGCSCConnectionSubsystem	Yes	GplusCommServer Primary	Name of Siebel HTTPSubSys, created for a primary <i>Gplus</i> Communication Server. See section “ <i>Creating a Connection Subsystem</i> ” in Chapter 8 of the <i>Gplus Adapter 7.5 for Siebel CRM Deployment Guide</i> .
BackupGCSCConnectionSubsystem	No		Name of Siebel HTTPSubSys, created for a backup <i>Gplus</i> Communication Server.
InteractionId	Yes, if ThirdPartyId is absent		Genesys interaction ID to be used to pull the interaction

Table 18: StopWorkItem Method Input Parameters (Continued)

Parameter	Required	Default Value	Description
ThirdPartyId	Yes, if InteractionId is absent		Siebel record ID corresponding to the Genesys interaction ID to be used to pull the interaction
InteractionField	No	Call Id	Field name used to store Genesys interaction ID
RecIdField	No	Id	Field name to be used as Siebel record ID
BusObject	No	Action	Business object name
BusComp	No	Action	Business component name
SuccessStatus	No		Value to be set to StatusField in the case where “stop processing” is successful
FailedStatus	No		Value to be set to StatusField in the case where “stop processing” failed
StatusField	No	Status	Field name used to store a record status
UserName	No		Agent ID. If it is set, the method will assign the record to this agent
UserField	No	Primary Owned By	The field name used to store the Agent ID which is owns the record

Route Method

The route method is used to send a route request to the Genesys environment. The route method gathers all the necessary input parameters and sends a route request to a *Gplus* Communication Server. If you would like to add some *AttachedUserData* values to a route request, you should add custom input arguments. All input arguments except the predefined arguments will be attached to a route request as *AttachedUserData*. The method returns *RouteResult* and *RouteMessage* parameters. A *RouteResult* value of 0 indicates

successful operation; a nonzero value represents an error code. The `RouteMessage` parameter contains the Genesys `InteractionId`.

For a list of input parameters and their descriptions, see [Table 19](#).

Table 19: Route Method Input Parameters

Parameter	Required	Default Value	Description
<code>PrimaryGCSCConnectionSubsystem</code>	Yes	<code>GplusCommServerPrimary</code>	Name of Siebel <code>HTTPSubSys</code> , created for a primary <i>Gplus</i> Communication Server. See section “ <i>Creating a Connection Subsystem</i> ” in Chapter 8 of the <i>Gplus Adapter 7.5 for Siebel CRM Deployment Guide</i> .
<code>BackupGCSCConnectionSubsystem</code>	No		Name of Siebel <code>HTTPSubSys</code> , created for a backup <i>Gplus</i> Communication Server.
<code>ConnectionName</code>	Yes		The Connection name (Application of Interaction Server) as specified in the Connections tab for the <i>Gplus</i> Communication Server. If you use this method as a communication profile command, you may skip the <code>ConnectionName</code> parameter, as this method will read the parameter value from communication profile parameters. But if you use this method in a workflow, you must set the parameter value.
<code>InteractionField</code>	Yes, only if you are using Pull/Stop functionality	<code>Call Id</code> (only for Siebel eMail media type)	Field name used to store Genesys interaction ID If this parameter is set, the method will write a Genesys interaction ID into this record field.

Table 19: Route Method Input Parameters (Continued)

Parameter	Required	Default Value	Description
InteractionType	No	Inbound	The Interaction type. Do not change the default value unless Genesys recommends it
InteractionSubtype	No	InboundNew	The Interaction subtype. Do not change the default value unless Genesys recommends it
SubmitQueue	Yes	-	The name of the queue where the interaction should be submitted.
StatusField	No	Status	Field name used to store a record status
SuccessStatus	No		Status of record to be set in the case where a route request was successful
FailedStatus	No		Status of record to be set in the case where a route request failed
ThirdPartyId	Yes	-	Record id for the routing interaction. This depends on the media type: for Siebel eMail it is Activity Id; for Siebel Service Request it is Service Request Id.
RefreshViewAfterUpdate	No	-	If this parameter has the value Yes and the method was called from the communication configuration command, the active view will be refreshed after the route request is sent to show the result record update.

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 communication configuration to mark an interaction as done and remove it from the list of active workitems.

If you use the Media Routing Component for routing an interaction 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 communication menu to mark an active interaction as done.

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

```
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` command, but the `MarkDoneQueue` parameter is not required, as a default queue `__STOP__` is used.

You may also invoke directly the `OpenMediaPlaceInQueue` method of the Adapter driver as follows:

```
var outQueue = "Mark_Done_Queue_Name";
var ctiSvc = TheApplication().GetCTIService();
var serv = TheApplication().GetService("MCR TopActiveWorkItem");
var inp = TheApplication().NewPropertySet();
var outp = TheApplication().NewPropertySet();
serv.InvokeMethod("GetTopActiveItem", inp, outp);
var Id = outp.GetProperty("DriverWorkTrackID");
var dataCTI = ctiSvc.GetCurrentCallData();
dataCTI.SetFieldValue("Queue", outQueue);
dataCTI.SetFieldValue("TrackingID", Id);
ctiSvc.InvokeCommandWithData( "OpenMediaPlaceInQueue", dataCTI);
```

Using the Media Routing Component for Routing Siebel Work Items

The Media 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). Please refer to the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide* for general information on configuration of the Media Routing Component.

Group buttons on the communication 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.

The following is a command sample. Please note that a media type is set with the prefix @ in the DeviceCommand parameter and in the FilterSpec parameter:

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

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

To send a route request, you must use a route method of the GplusMediaRoute Business Service. The following is a sample route command to route a service request. To use it, please set the SubmitQueue parameter. If you send a route request in a workflow, you must also set the ConnectionName parameter value. Please refer to “Route Method” on [page 71](#).

```

[Command:SendRouteSR]
  Description    = "Route Service Request"
  Title          = "Route Service Request"
  ServiceMethod = "GplusMediaRoute.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 OpenMedia"
  CmdData       = "SendRouteSRcmd"

[CmdData:SendRouteSRcmd]
  RequiredField.SR Number      = "?*"
  ServiceParam.Subject         = "{Abstract}"
  ServiceParam.BackupGCSCONNECTIONSubsystem =
    "GplusCommServerPrimary"
  ServiceParam.BusComp         = "Service Request"
  ServiceParam.BusObject       = "Service Request"
  ServiceParam.PrimaryGCSCONNECTIONSubsystem =
    "GplusCommServerPrimary"
  ServiceParam.InteractionField = "Integration Id"
  ServiceParam.RecIdField       = "SR Number"
  ServiceParam.RoutingMediaType = "ServiceRequest"
  ServiceParam.SR_Type          = "{SR Type}"
  ServiceParam.SubmitQueue      = "CHANGE ME"
  ServiceParam.ThirdPartyId     = "{SR Number}"

```

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 `ThirdPartyId`. Please note that the `InteractionField` parameter is set in this example, so it will be possible to use Pull/Stop functionality. As the `SuccessStatus` parameter is absent, the method will not update the record status. If you want to do a record status update, you may add `SuccessStatus` and `FailedStatus` parameters. Please see “Route Method” on [page 71](#).

In this sample, the `ServiceParam.Subject` is an optional parameter, and it will be 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 please refer to “Using the *GplusMediaRoute* Business Service” on [page 66](#).

To accept a routed work item, the agent must click the Accept group button or the Accept Multimedia Interaction subbutton. You must create an Event Handler for the `OpenMediaAccepted` event for each custom media type to open a Siebel view for the routed work item.

Also, you should set the `QuerySpec` parameter to the value `FieldName='{ThirdPartyId}'` where `FieldName` is the ID field for a Siebel work item. Set the `SingleView` parameter to a proper Siebel view name. See a sample below:

```
[EventHandler:OpenMediaSelectedSR]
Filter.MediaType = "ServiceRequest"
Profile = "Gplus OpenMedia"
Comments = "EventHandler samples for Siebel work items routing"
Order = "50"
Response = "OpenSiebelSRView"
DeviceEvent = "OpenMediaSelected"
```

```
[EventResponse:OpenSiebelSRView]
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 like this:

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

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

```
[EventLog:EventLogAcceptSR]
BusComp = "Action"
BusObj = "Action"
LogField.Owner = "{@UserName}"
QuerySpec = " SR Number =' {ThirdPartyId} '"
```

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 “Applet Customization” on [page 74](#).

The provided 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.

To try the sample, please remove the comment marks in the `OBJECTS/GenComm7_universal.def` file before import of a configuration and perform the above-mentioned actions. Create a custom media type in the Genesys environment and then add in the `Channel String` parameter of the *Gplus* OpenMedia profile driver. Set the proper values for the command and events used. For more information about the Media Routing Component deployment and configuration, please refer to the *Gplus Adapter 7.5 for Siebel CRM Deployment Guide*, Chapter 8.



Chapter

7

Using Siebel Data from the Genesys Universal Routing Solution

It may be beneficial to use 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 (SOAP) interface. On a high level, 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:

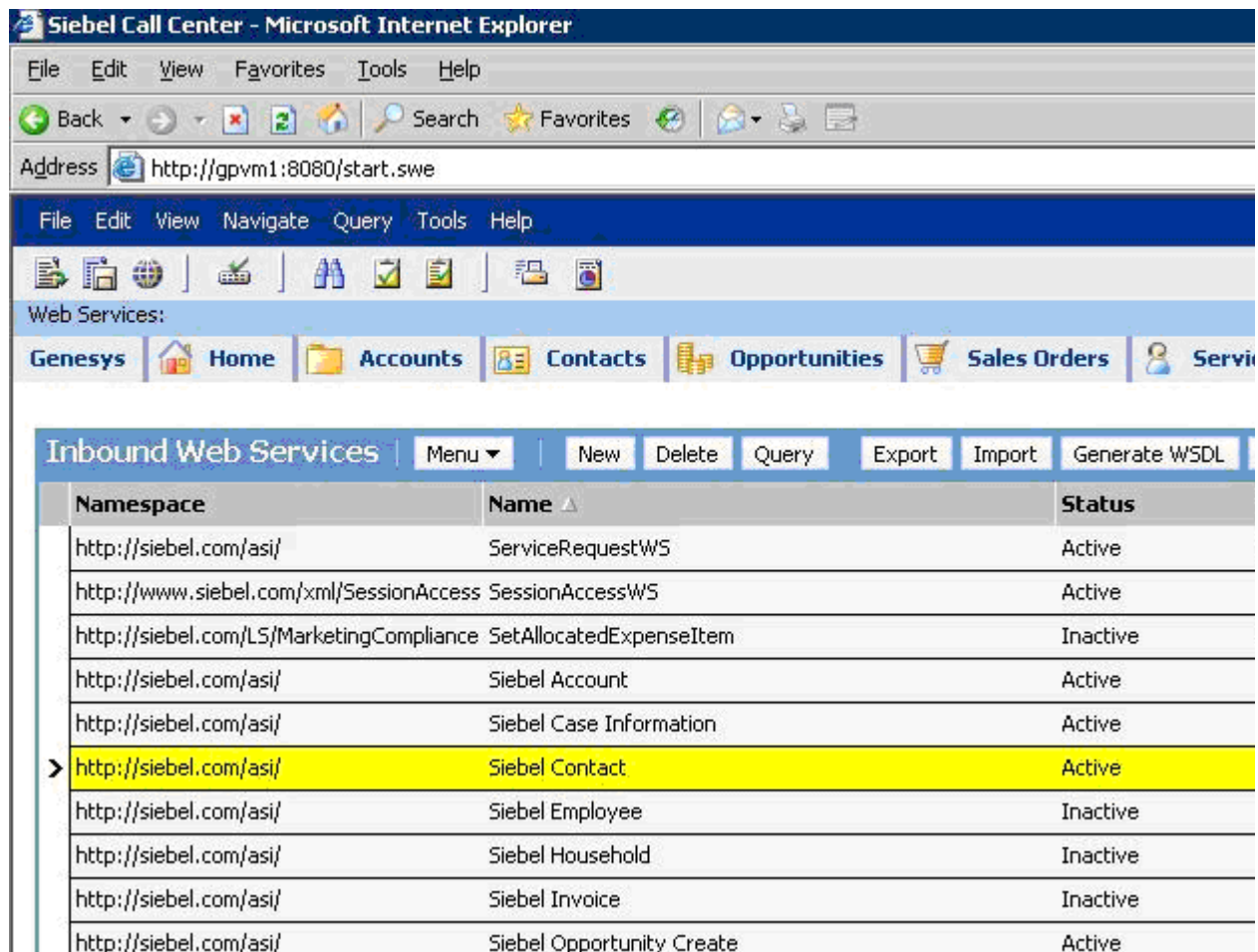
- [Checking the Inbound Web Service, page 79](#)
- [Using the Web Service in Genesys Universal Routing, page 80](#)

Checking the Inbound Web Service

To administrate Siebel Inbound Web Services on the Siebel Client:

1. Navigate to View > Site Map > Administration - Web Services > Inbound Web Services. Refer to Figure 3 on [page 80](#).
2. Select a web service. (This chapter uses the standard Siebel Contact web service as an example.)

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



Namespace	Name	Status
http://siebel.com/asi/	ServiceRequestWS	Active
http://www.siebel.com/xml/SessionAccess	SessionAccessWS	Active
http://siebel.com/LS/MarketingCompliance	SetAllocatedExpenseItem	Inactive
http://siebel.com/asi/	Siebel Account	Active
http://siebel.com/asi/	Siebel Case Information	Active
> http://siebel.com/asi/	Siebel Contact	Active
http://siebel.com/asi/	Siebel Employee	Inactive
http://siebel.com/asi/	Siebel Household	Inactive
http://siebel.com/asi/	Siebel Invoice	Inactive
http://siebel.com/asi/	Siebel Opportunity Create	Active

Figure 3: Siebel Inbound Web Services Administration View

Using the Web Service in Genesys Universal Routing

On the Genesys side, include the Web Service object into an appropriate routing strategy to perform (SOAP) requests to the Siebel database. Refer to Figure 4 on [page 81](#).

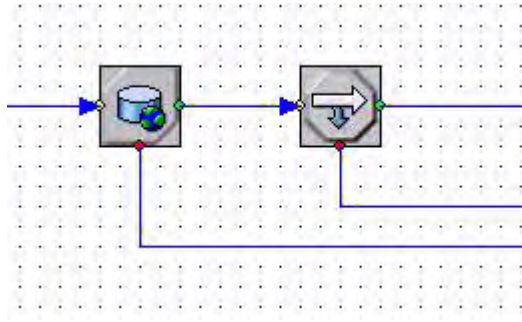
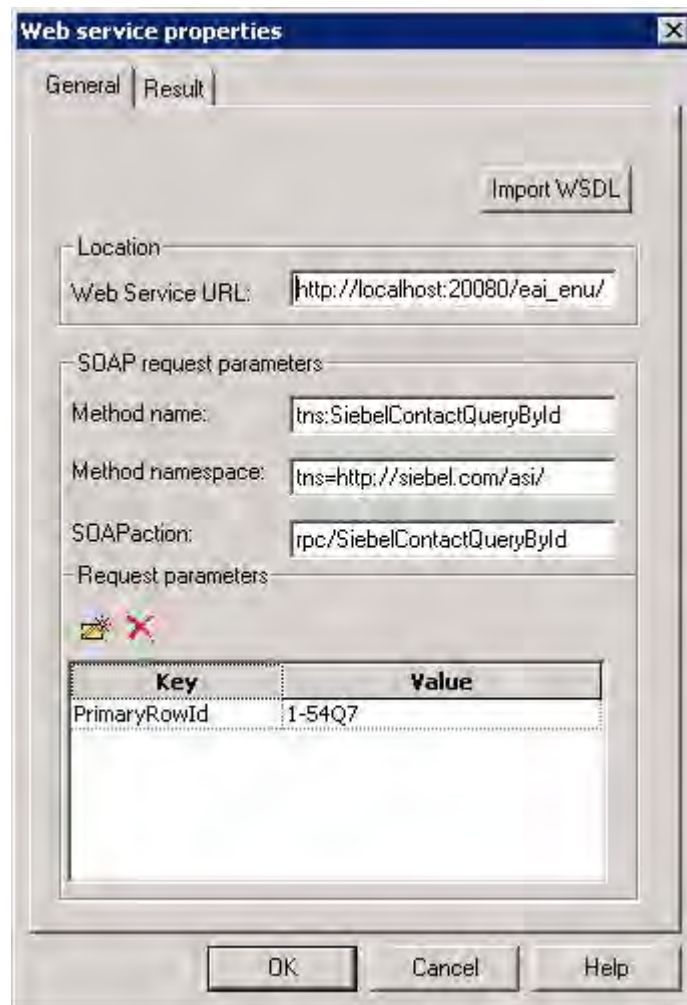


Figure 4: Example of Using the Web Service Object in a Routing Strategy

The Web Service object has the following properties (refer to Figure 5 on [page 82](#)):

- The Web Service URL contains the URL to the appropriate web service. For our example it has the following structure (you can get it from the Address field of the Service Port applet of the Inbound Web Services Administration view):
`http://<webserver>/eai_<lang>/start.swe?SWEExtSource=WebService&SWEExtCmd=Execute&UserName=<UserName>&Password=<Password>`
- The Method name contains the SOAP method name to invoke. It has the following structure:
`<namespace variable name>:<SOAP method name>`
 You can find 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:
`<namespace variable name>=< namespace>`
- The SOAPaction contains the SOAP action. It has the following structure:
`rpc/< SOAP method name >`
- Request parameters contains key-value pairs which correspond to the SOAP method input parameters.



The image shows a 'Web service properties' dialog box with a 'General' tab selected. It contains fields for 'Web Service URL', 'Method name', 'Method namespace', 'SOAPAction', and a table for 'Request parameters'. The 'Web Service URL' is 'http://localhost:20080/eai_enu/'. The 'Method name' is 'tns:SiebelContactQueryByld', 'Method namespace' is 'tns=http://siebel.com/asi/', and 'SOAPAction' is 'rpc/SiebelContactQueryByld'. The 'Request parameters' table has one row with 'PrimaryRowId' as the key and '1-54Q7' as the value.

Key	Value
PrimaryRowId	1-54Q7

Figure 5: Web Service Object Properties - General Tab

To get the method's parameter list, refer to the Siebel Tool 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. (Refer to Figure 6 on [page 83](#).)



Figure 6: Web Service Object Properties - Result Tab

Reference to the SOAP response generally depends on the requested method output variables type. For our example, reference to the EmailAddress field is: `SiebelContactQueryByIdResponse.SiebelMessage.ListOfContactInterface.Contact.Contact.EmailAddress`

For more information related to 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 (refer to Figure 4 on [page 81](#) and Figure 7 on [page 84](#)).

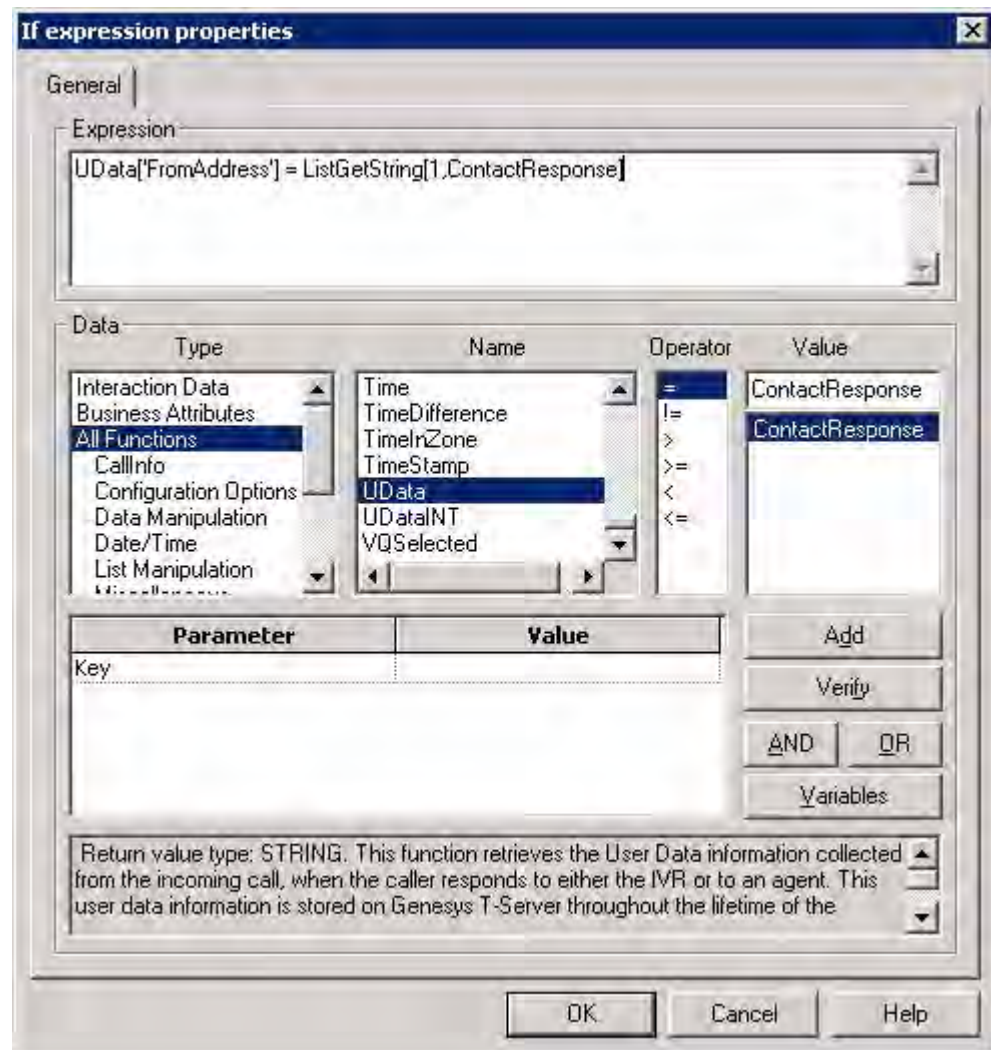


Figure 7: “If” Expression Object Parameters

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

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

Note: 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.



Index

A

Applet Customization. 74

B

brackets
 angle. 10

C

Campaign Lists
 exporting to Genesys 19
Campaign Synchronizaton
 data flow. 19
Campaign Synchronization. 15, 49
 data flow. 13, 19
 request types 29
Campaign Synchronization Response . . . 34
CampaignInfo 29

D

Data Flow 13, 19
 sample. 22
document
 version number 9
document errors
 commenting on 10
documentation
 Library CD. 11
 ordering 11

F

Format Configuration Guidelines. 40
Framework 6

G

Genesys
 Documentation Library CD 11
GplusMediaRoute 66
GetTopWorkItem 66
MarkWorkItemDone. 66
PullInteraction. 69
 route 71
StopWorkItem. 70

I

Inbound XML Schema 26

L

List Import 14, 39
ListInfo. 31

M

Media Routing Component Customization . . 65
Method Descriptions. 50
Method Usage Guidelines 50
migration. 11

O

Outbound Contact Server 6

R

RecordInfo. 34
Request 29
Request Types. 29
Routing Siebel Work Items 74



S

Script Example	56
Siebel Configuration Guidelines	39
Style Sheet Configuration Example	45
Style Sheet Configuration Guidelines	43
symbols	
<>	10
Synchronization Summary Format	59
Synchronization Summary Usage	59

T

Technical Support website	11
typographical styles	9

U

Using Siebel Data from the Genesys Universal Routing Solution.	79
Using Synchronization Summary	61

V

version numbering	
document	9

W

Web Service	79
checking inbound	79
using in URS	80
Web Service object	81