Configuration SDK 7.6

Web Services

Developer's Guide

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 DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

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

Your Responsibility for Your System's Security

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

Trademarks

Genesys, 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)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	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: 76sdk_dev_conf-ws_09-2008_v7.6.101.00

Table of Contents

Preface

Welcome to the *Configuration SDK 7.6 Web Services Developer's Guide*. This guide introduces you to the Configuration API (application programming interface) architecture, provides deployment planning information including a list of supported toolkits, and lists code examples in Java and C# that show how to access Genesys configuration information.

This guide is valid only for the 7.6 release(s) of this product.

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

This preface provides an overview of this guide, identifies the primary audience, introduces document conventions, and lists related reference information:

The Configuration SDK (software development kit) Service provides you with access to the Configuration API of the Genesys Integration Server (GIS). This API enables you to access and update configuration object information from your own web-based client application.

The Configuration SDK Service also includes developer documentation and code examples to help you understand the API's functionality and how to write an application that interacts successfully with the GIS Configuration API.

Intended Audience

This guide, primarily intended for developers, assumes that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

You should also be familiar with these tools:

- XML Schema
- XPath
- SOAP
- WSDL (Web Services Description Language)
- Depending on the technology choice for client development, working knowledge of Java or of Microsoft C#.

Developers should be familiar with the Genesys Framework, especially with the Configuration Layer (referred to as "Configuration Management Environment" or "CME" in earlier Genesys product documentation).

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's 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, this guide contains the following chapters:

- Chapter 1, provides an overview of the SDK Service and its architecture.
- Chapter 2, introduces the code examples and explains how to install, compile, and run them.
- Chapter 3, includes three examples of how to use the Session Service to log in, verify licensing, and log out.
- Chapter 4, introduces the first step in connecting to the Configuration SDK Service and then releasing the connection at the end of a session.
- Chapter 5, shows how you might use the `get` method to retrieve configuration information from the Configuration Database.
- Chapter 6, provides an example of how to make modifications to the configuration database, such as additions and deletions, and changes to attributes such as agent skills.
- Chapter 7, shows how to register for specific information and then send requests to receive updates using either the Polling or Blocked notification.
- Chapter 8, provides an example of how to construct a client server that can receive automatic updates from GIS concerning configuration objects to which you have subscribed.
- Chapter 9, includes three examples of how to use the Session Service to log in, verify licensing, and log out.
- Chapter 10, introduces the first step in connecting to the Configuration SDK Service and then releasing the connection at the end of a session.
- Chapter 11, shows how you might use the `get` method to retrieve configuration information from the Configuration Database.
- Chapter 12, provides an example of how to make modifications to the configuration database, such as additions and deletions, and changes to attributes such as agent skills.
- Chapter 13, shows how to register for specific information and then send requests to receive updates using either the Polling or Blocked notification.
- Chapter 14, provides an example of how to construct a client server that can receive automatic updates from GIS concerning configuration objects to which you have subscribed.

Document Conventions

This document uses some stylistic and typographical conventions with which you might want to familiarize yourself.

Document Version Number

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

76fr_ref_02-2008_v1.00

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 emphasis, for documents' titles, for definitions of (or first references to) unfamiliar terms, 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 looks like `teletype or typewriter text`, 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.

Monospace is also used for any text that users must manually enter during a configuration or installation procedure, or on a command line:

Example:

- Enter `exit` on the command line.

Screen Captures Used in This Document

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

Square Brackets

Square brackets indicate that a particular parameter or value is optional within a logical argument, a command, or some programming syntax. That is, the parameter's or value's presence is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. Here is a sample:

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

Angle Brackets

Angle brackets indicate a placeholder for a value that the user must specify. This might be a DN or port number specific to your enterprise. Here is a sample:

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

Related Resources

Consult these additional resources as necessary:

- The *Configuration SDK 7.6 Web Services API Reference,* which details the messages, operations, data types, and constants as defined in the Configuration WSDL file that governs communication between the client application and GIS.

- The *Statistics SDK 7.6 Web Services Developer's Guide,* which explains how to set up the Statistics SDK code examples and presents the text of the examples along with explanatory comments.
- The *Genesys Integration Server 7.6 Deployment Guide,* which provides installation, configuration, and starting and stopping instructions for GIS.
- The *Statistics SDK 7.6 Web Services API Reference,* which details the messages, operations, data types, and constants as defined in the Statistics WSDL file that governs communication between the client application and GIS.
- *Framework 7.x Configuration Manager Help* for assistance in using Configuration Manager.
- The *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- The *Genesys Migration Guide*, also on the Genesys Documentation Library DVD, which provides a documented migration strategy from Genesys product releases 5.1 and later to all Genesys 7.x releases. Contact Genesys Technical Support for additional information.
- The Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at http://genesyslab.com/support.

Information on supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys 7 Supported Operating Systems and Databases*
- *Genesys 7 Supported Media Interfaces*

Genesys product documentation is available on the:

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

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 About the Configuration SDK Service

The Genesys Configuration SDK Service product provides you with all of the elements and resources, including documentation and code examples, required to assist you in developing an application that allows users to view and update configuration information.

This chapter presents the following topics:

Configuration SDK Service Overview

The Configuration SDK Service consists of the following elements:

Genesys Integration Server (GIS)—Exposes the Session and Configuration Services.

- GIS is a web application embedded in either the Tomcat web container or the WebSphere web container. It functions as a server in relation to your client application.
- GIS and your client application communicate using a request/response message flow.
- Each message consists of an XML message body that is sent via HTTP and wrapped using the programming language of your choice. The examples included in this SDK Service are in Java and C#.

The Session Service—Provides an interface to Configuration Server for simple user name and password verification, and license authentication.

- Session Service methods are documented in this guide and also in the *Configuration SDK 7.6 Web Services API Reference.*

Note: You do not need a license for the Session Service.

The Configuration SDK Service—Enables you to retrieve and update configuration information. Your application directs messages for the Configuration SDK Service to GIS, which performs license control for each message and then sends the messages to the Configuration Server or CS Proxy SOAP interface you have configured.

Note: For details on understanding and deploying Configuration Server and CS Proxy, see the *Framework 7.x Deployment Guide*.

Architecture

Figure 1 illustrates the relationships between your client application, GIS, and the Genesys Framework.

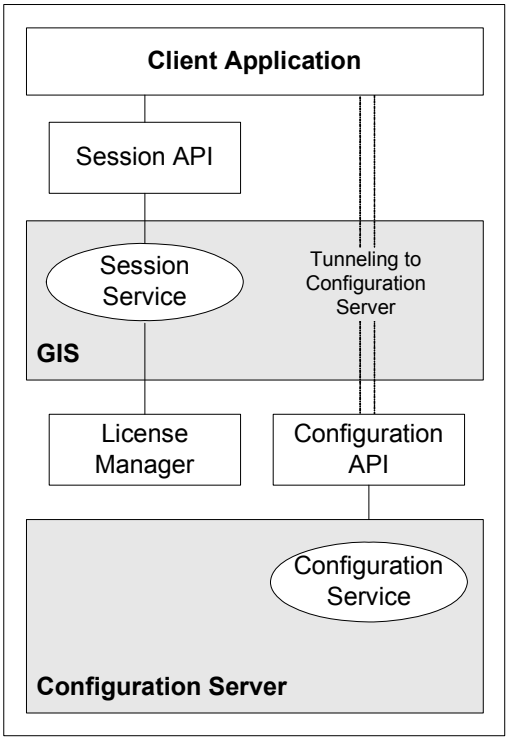

Figure 1: GIS Framework Architecture

Note: Depending on the licenses you purchase, GIS might offer additional services that are not documented or illustrated here. For information about other SDK services that GIS provides, see the "Introducing GIS Services" chapter of the *Genesys Integration Server 7.6 Deployment Guide.*

Interacting with Genesys Framework Components

GIS works with the Genesys Framework components described in the following subsections.

License Manager

GIS uses the Session Service to authenticate users with the License Manager, and to authorize access to Configuration Server information.

Configuration Server/CS Proxy

GIS uses the Configuration Server SOAP API to access configuration information. This SOAP interface can receive messages that are sent by your client application and passed on by GIS.

Your Configuration Server can be configured as either a master Configuration Server or a CS Proxy. In either case, you must set the SOAP interface option in the Configuration Server Application object.

Note: If you are using Configuration Server 6.5, the SOAP interface is only available if your Configuration Server is configured as a CS Proxy.

Management Layer

You can use the Management Layer to view, start, or stop GIS from Solution Control Interface (SCI). To do so, you must also run Local Control Agent (LCA) on the host that supports GIS to monitor activity on the local machine and communicate with Solution Control Server (SCS).

Note: Management Layer is not supported if you integrate GIS with WebSphere.

Supported Versions of Genesys Framework

The Genesys Configuration SDK Service requires Configuration Server 7.*x* or CS Proxy 6.5.

Communication Technologies

GIS presents Genesys resources to your client application through a set of request and response operations that use the communication protocols displayed in Figure 2.

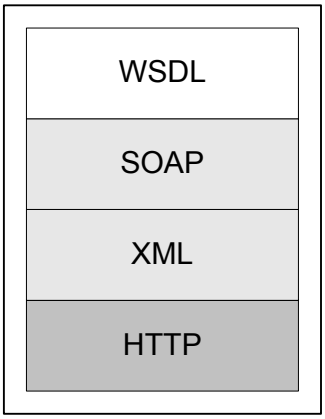

Figure 2: GIS Communication Protocols

Your client application uses HTTP and TCP/IP to connect to the GIS web server and transmit XML-based messages using the SOAP protocol. The message payload conforms to the WSDL specification.

HTTP—The underlying transport protocol. GIS is compatible with HTTP versions 1.0 and 1.1.

XML—The underlying language used to define elements of the Session Service and the Configuration SDK Service.

SOAP—A simple XML-based protocol that carries messages defined by the WSDL file. Using SOAP ensures that the interface is language-, platform-, and vendor-neutral while also providing easy access and integration with other Genesys components.

WSDL (Web Services Description Language)—Defines message parameters between the client application and the web server, in this case GIS. These communication conventions for the Configuration SDK Service are defined in the Session and Configuration WSDL files.

The Configuration SDK Service uses two WSDL files, one each for the Session Service and the Configuration SDK Service.

Note: For more information on SOAP, WSDL, and other XML-based technologies, see `http://www.w3schools.com/default.asp` for tutorials and links.

Authentication and Authorization

Configuration Server implements a dual authentication scheme based upon the required `application` parameter in the `Register` request.

If the `application` parameter specifies a daemon-type application, such as `ThirdPartyServer`, then the Configuration Server performs authentication automatically using the account specified for that daemon application in the Application object. Otherwise, the user name and password of the person to be authenticated must be passed in the `Register` request to verify the identity of the client.

Note: If the specified application is a daemon-type application, then Configuration Server ignores any user name and password parameters entered.

Once verified, Configuration Server uses the account detected during the authentication process (either detected explicitly through a specified user name and password, or automatically through the daemon application's property) to check the client's access rights. These access rights include permissions to read, change, create, and remove objects during subsequent requests within the established session.

Message Flow

Because HTTP is stateless, all communication (including logging in) is performed as a set of request/response operations. To get a specific piece of configuration data, the client sends a request for that information about a particular object and then receives a response from GIS.

All communication follows a modular request/response pattern: input messages are sent by the client application and output messages are returned by GIS. Figure 3 shows the request/response pattern.

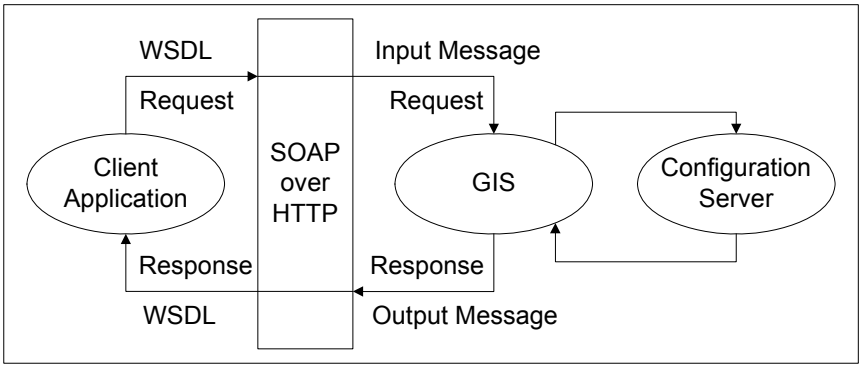

Figure 3: GIS Request/Response Pattern

The stages of the data flow are described in the following steps:

1. The client application initiates a session with GIS with a `login` operation.

   The client opens a session to GIS by sending a `login` message with a user name and password. The Session Service registers this user with Configuration Server, and enables licensing authentication.

   GIS returns a `login` response your client application with the Session ID, which must be attached to all subsequent messages for this transaction. This session remains open until the configured timeout expires without any client-server interactions. (You can use the `session_lifespan` option in the Configuration Server Application object to adjust this timeout value.)

2. GIS responds.

   The response content is either an acknowledgment that the request is now active and that GIS is beginning to retrieve the data or an error message indicating a problem.

3. The client application must now subscribe to the GIS_CONFIGSERVICE license using the `getService` method.

4. GIS checks with License Manager and then responds.

   The response content is either the license to which the client subscribed or an error message indicating a problem with the license.

5. The client application can request (`get`) information on an object to see the current status of that object, make changes to (`update`) objects, or `subscribe` to an object for ongoing updates over time.

6. To close a session, your client application sends a logout message to the Session Service. GIS then returns the logout response.

> **Note:** You do not log out of the Configuration service. Your session ends at the expiration, without client-server interactions, of the timeout period you set using the `session_lifespan` option in the Configuration Server Application object.

Figure 4 on page 21 shows the major elements of the data flow between your client application, which initiates transactions in most circumstances, and GIS. It also shows the communication between GIS and whichever Framework component is relevant for each specific request.

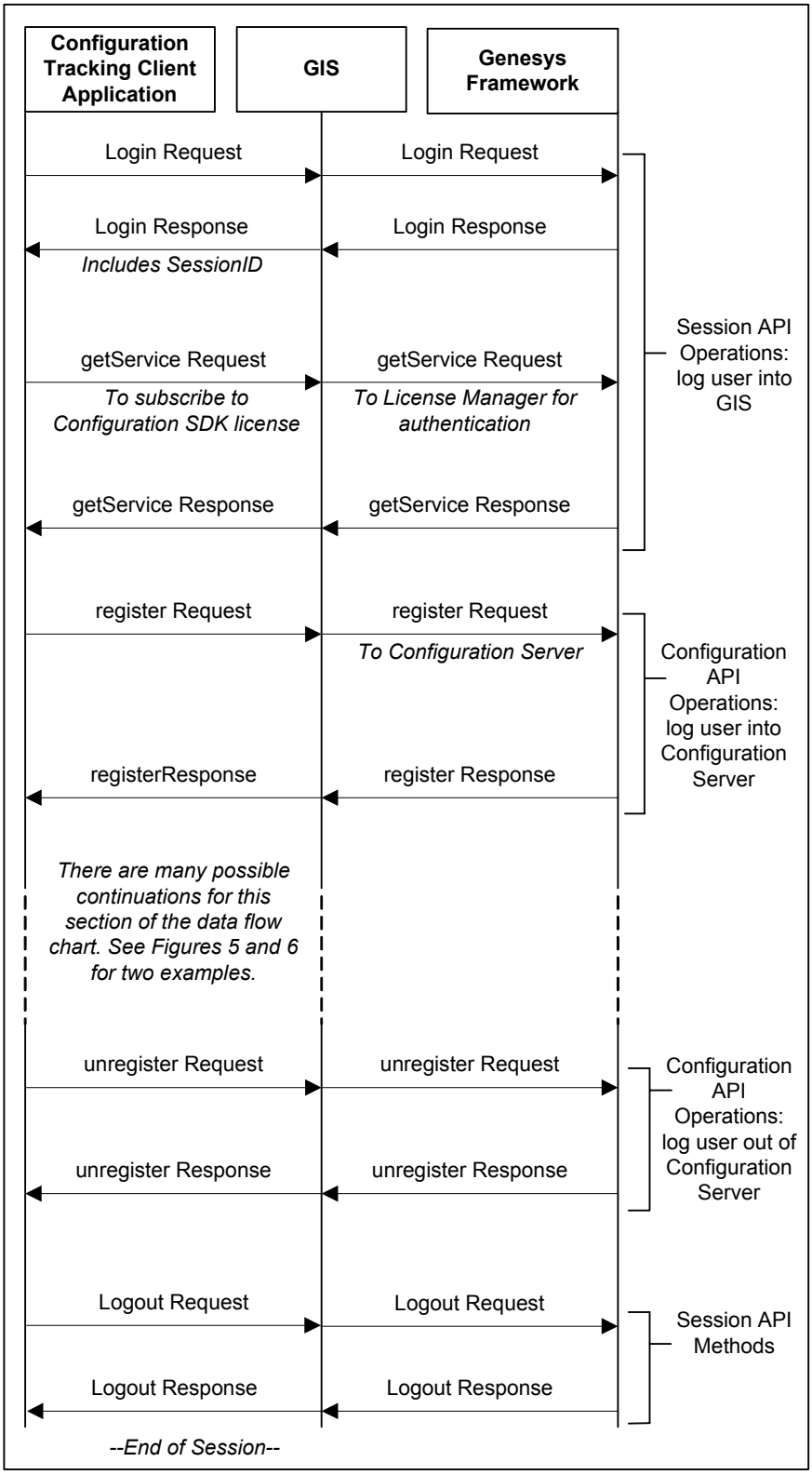

Figure 4: Configuration SDK Service Data Flow

Figure 5 shows an example of the request/response sequence you might use in the `Update` operation. For a discussion of the notification modes you can use with the subscribe operations, see "Notification Modes".

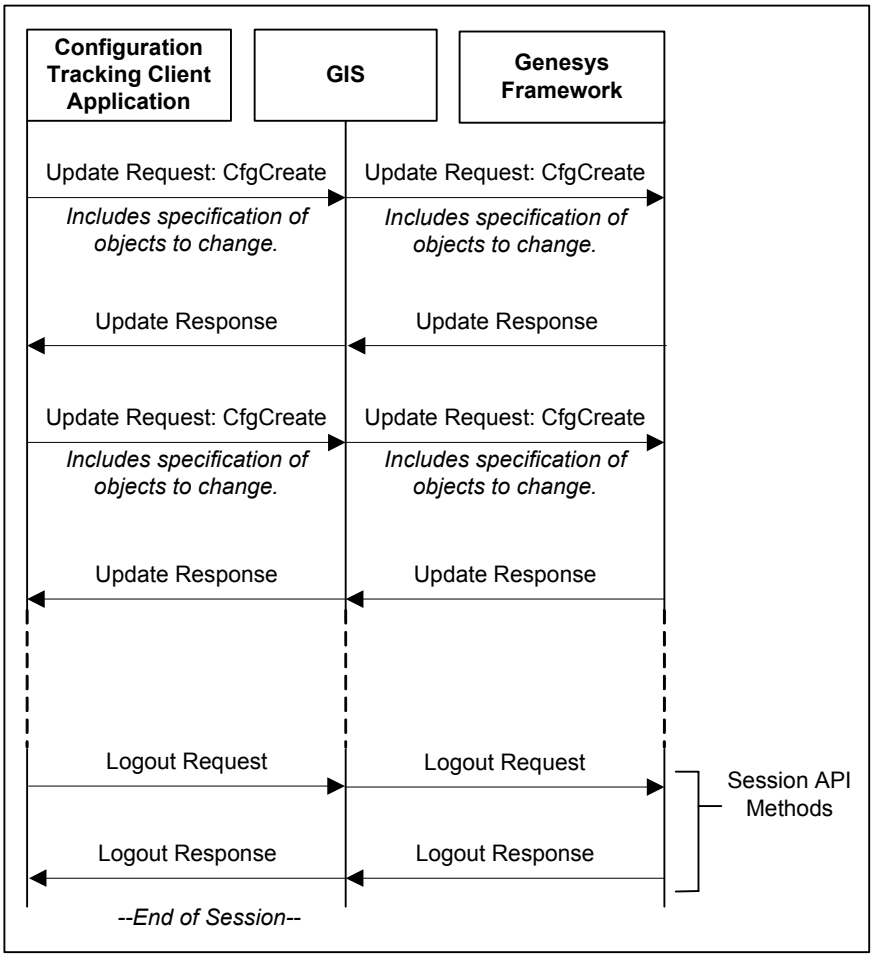

Figure 5: `Update` Data Flow

Notification Modes

The Configuration interface allows users to subscribe to notifications regarding changes to Configuration Database objects using either the `subscribe` message for simple subscriptions, or the `subscribeEx` message which allows more complex request types.

To retrieve subscribed data, your client application must either:

- Implement the Unsolicited Notification feature. To use this feature, you specify a URL to which configuration updates are sent automatically.
- Request notification messages from the server using `getEvent`. The server does not respond immediately, but sends updated data when the change occurs that corresponds to your subscription.

Unsolicited Notification

The Unsolicited Notification method in the Java and C# configuration examples sends a request to GIS for automatic updates to a configuration object.

If you are using Java, this process requires a server on the client side, because GIS can only send data to a web server. The client server must be able to listen for and process GIS-specific SOAP messages. The example in Chapter 8 includes a variant of a simple HTTP server.

The Unsolicited Notification method in the C# configuration example also sends a request to GIS for automatic updates on a configuration object to which you have already subscribed. The example in Chapter 14 shows how to use the Remoting feature available in Microsoft .NET to receive update messages from GIS.

Solicited Notification

Subscribing vs. Getting

`Subscribe` and `subscribeEx` differ from `get` and `getex` in the way in which your client application receives data.

- `Get` and `getex` retrieve immediate responses containing the values for the requested objects at the time of your request.
- `Subscribe` and `subscribeEx` enable you to tell the server what data you want and then use the `getEvent` message to prompt a response concerning the subscribed data. The server waits until the specified type of change occurs and then sends updated values.

There are two ways for the client to set up a communication channel for notification:

- **Asynchronous getEvent Requests**—The client calls a `getEvent` request asynchronously, assigns a dedicated callback function to receive the response, and then continues to send requests synchronously. When the client receives a response to the callback function, it processes the notification and then repeats this process to receive the next notification.
- **Dedicated Connection**—The client opens a separate connection using a thread dedicated to processing notifications. The client registers using this connection, makes subscriptions, and calls `getEvent` requests in a synchronous loop.

Figure 6, shows an example of the sequence of requests and responses that are used in the `subscribeEx` operation.

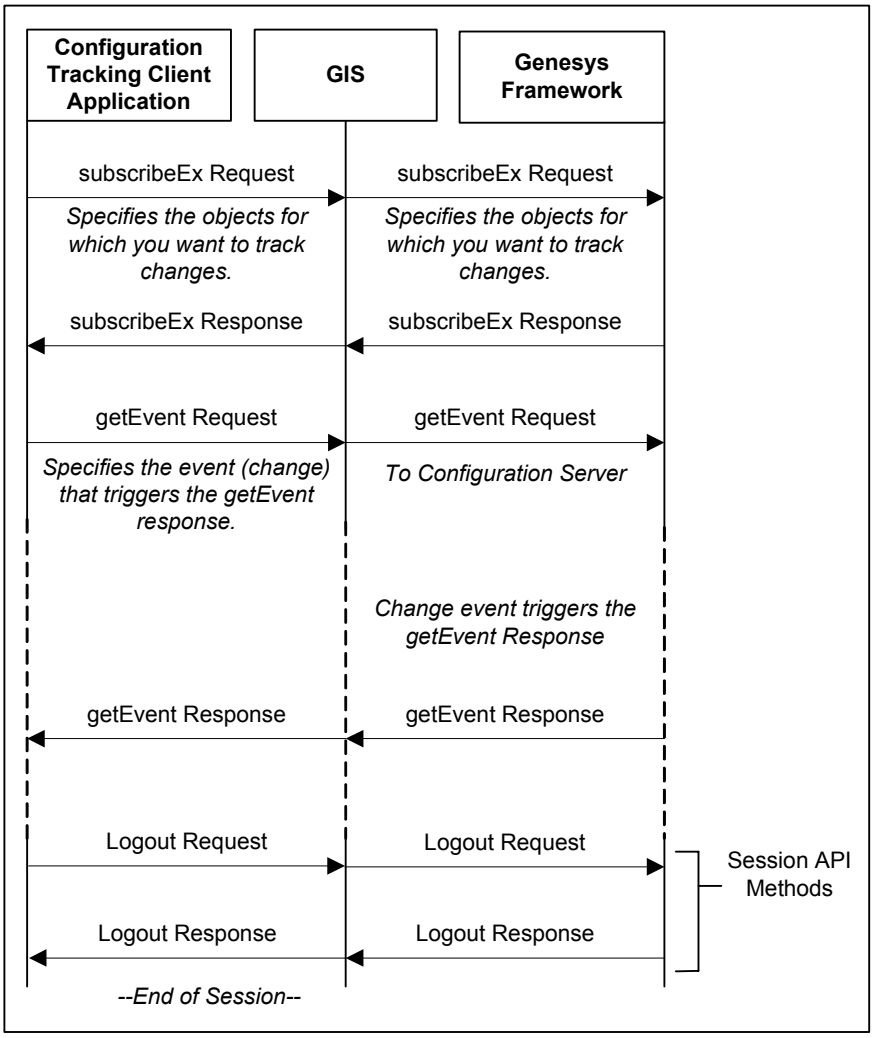

Figure 6: `SubscribeEx` Data Flow

About the Code Examples

The Configuration SDK Service includes short, compilable code examples, which are located on the DevZone Genesys Developer Documentation portal (`http://www.genesyslab.com/developer`), and on the Genesys Developer Documentation CD. They are intended to demonstrate how to exercise key functions and are provided in both Java and C#.

- Chapter 2 contains detailed information on the content of the examples, as well as instructions for installing and configuring them.

- Discussion of the Java code examples starts with Chapter 3, which presents the code examples specific to the Session Service, and continues through Chapter 8.
- Discussion of the C# code examples starts with Chapter 9, which presents the code examples specific to the Session Service, and continues through Chapter 14.

Note: These code examples are as accurate as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

New In This Release

This section summarizes the changes between Configuration SDK 7.6 Web Services and prior releases of this products. It identifies new options and compatibility issues.

New Default Value for Users Accounts

Starting with release 7.6, Configuration Server does not automatically assign new users to any `Access Groups`. Each new user:

- Is created with no privileges.
- Cannot log in to any interface.
- Cannot use a daemon application.

Each new user must be added to an `Access Group` by an Administrator or a user with permissions to update user accounts. For details, see the chapter "No Default Access for New Users" in the *Genesys 7.6 Security Deployment Guide*.

Chapter

2 About the Examples

Documentation for the Configuration SDK Service includes fully functional code examples in Java and C#. These code examples provide a model of the development process for your Genesys Integration Server (GIS) based configuration administration client application.

This chapter provides an overview of the examples, explains what each does, and describes the installation and configuration procedures required before you can run the examples.

The examples themselves appear, with explanations, in subsequent chapters. Java examples begin in Chapter 3 and continue through Chapter 8; C# examples begin in Chapter 9 and continue through Chapter 14.

This chapter contains the following sections:

Examples Overview

All code example packages included with the Configuration SDK Service can be found on the Genesys Developer Documentation DVD, or located through the Genesys DevZone Developer portal (http://www.genesyslab.com/developer).

The code examples show how to create session and configuration requests. Session requests include logging in users, logging out users, and license validation. Configuration requests include actions such as retrieving or updating information about configuration objects, such as Agents, Agent Groups, and Places.

Most of the examples build upon each other, with later examples using the functionality from earlier examples. For example, instead of repeating the code required to log in and create a session, later examples refer to the Session service examples that exercise those functions. Each example also includes a stand-alone option, which allows you to see how that particular feature is implemented.

You are strongly advised to run these examples in order to fully understand how they work and relate to one another.

Note: The content contained in return messages for this document will not reflect the exact content you will receive. Each customer site will return different results. However, the samples show the format and type of information to expect.

These examples are as accurate and complete as possible. However, they are not intended to be more than models for development. They do not combine to form a fully-functional client application. Genesys does not guarantee the performance of these examples or provide support for them.

Java Examples

These examples use Java to demonstrate the major functions you can perform using the Session Service and the Configuration SDK Service. Depending on the purpose of your client application, you must implement some (if not all) of the functionality shown in the examples.

About the Session Service Examples

The Session Service is a GIS service that logs in users from your client application and checks that they have a valid license for the Configuration SDK Service. You do not need a separate license to use the Session Service. After you have checked out a Configuration SDK Service license using the Session Service, your client application can send requests to the Configuration SDK Service.

The Session ID

The Session Service returns the Session ID in the response to your login request. This Session ID is a required part of all subsequent requests sent by your client application to the Configuration SDK Service, and marks the various request and response messages exchanged as members of the same transaction.

Session Example Files

There are three session example files:

- `CreateSessionExample.java`—Establishes a session with GIS. Used widely in the Configuration examples.
- `ConnectSessionServiceExample.java`—Checks out a service license.
- `IdentifyServicesExample.java`—Browses through and returns a list of the available services or releases service licenses.

All of the session example files are located in the `com.genesyslab.gis.services.session` package. For details about these examples, see Chapter 3.

About the Configuration Service Examples

To view or update a configuration object, your application must send a request for information or instructions on what changes to make, and then analyze the response.

The configuration examples demonstrate how to register with Configuration Server, get information on configuration objects, add, delete and change configuration objects, and request unsolicited notification of configuration changes from GIS. Review the code examples to learn how to create the dependent or necessary objects to accomplish these tasks.

Configuration Service Example Files

There are six configuration example files included with this product:

- `RegisterToConfigurationServiceExample.java`—Demonstrates how to register for a Configuration service, release the service, and log out from GIS. See Chapter 4 for details.
- `GetConfigurationInformationExample.java`—Uses the `get` method to obtain configuration information. See Chapter 5 for details.
- `UpdateConfigurationDataExample.java`—Makes additions, deletions, and changes to configuration data using the `update` method. See Chapter 6 for details.
- `SolicitedNotificationByBlockingExample.java`—Subscribes to configuration information and then retrieves updates using the Blocking notification method. See Chapter 7 for details.
- `SolicitedNotificationByPollingExample.java`—Subscribes to configuration information and then retrieves updates using the Polling notification method. See Chapter 7 for details.
- `UnsolicitedNotificationExample.java`—Subscribe to configuration information and then retrieves updates using the Unsolicited Notification method. See Chapter 8 for details.

All configuration examples are located in the `com.genesyslab.gis.services.configuration` package.

Example Presentation

The remaining chapters in this document focus on one example, listing the code in text format and adding commentary to help you understand the example. You can compile and run these examples to demonstrate basic SDK Service functionality and to help you plan your client application development.

Each example in the following chapters will include:

- A short introduction to the functionality.
- The example code broken into sections, with commentary.
- A stand-alone version of the example, including a sample response.

C# Examples

These examples use C# to demonstrate the major functions you can perform using the Session Service and the Configuration SDK Service. Depending on the purpose of your client application, you must implement some (if not all) of the functionality shown in the examples.

About the Session Service Examples

The Session Service is a GIS service that logs in users from your client application and checks that they have a valid license for the Configuration SDK Service. You do not need a separate license to use the Session Service. After you have checked out a Configuration SDK Service license using the Session Service, your client application can send requests to the Configuration SDK Service.

The Session ID The Session Service returns the Session ID in the response to your login request. This Session ID is a required part of all subsequent requests sent by your client application to the Configuration SDK Service, and marks the various request and response messages exchanged as members of the same transaction.

Session Example Files

There are three session example files:

- `CreateSessionExample.cs`—Establishes a session with GIS. Used widely in the Configuration examples.
- `ConnectSessionServiceExample.cs`—Checks out a service license.

- `IdentifyServicesExample.cs`—Browses through and returns a list of the available services or releases service licenses.

All of the session example files are located in the `GISServices` directory. For details about these examples, see Chapter 9.

About the Configuration Service Examples

To view or update a configuration object, your application must send a request for information or instructions on what changes to make, and then analyze the response.

The configuration examples demonstrate how to register with Configuration Server, get information on configuration objects, add, delete and change configuration objects, and request unsolicited notification of configuration changes from GIS. By reviewing the example code, you will learn how to create the dependent or necessary objects to accomplish these tasks.

Configuration Service Example Files

There are six configuration example files included with this product:

- `RegisterToConfigurationServiceExample.cs`—Registers for a Configuration service, then releases the service and logs out from GIS. See Chapter 10 for details.
- `GetConfigurationInformationExample.cs`—Demonstrates how to accomplish Configuration Service [get] requests, demonstrating different kinds of XPath expressions and the processing of these queries' results. See Chapter 11 for details.
- `UpdateConfigurationDataExample.cs`—Uses the `update` method to make additions, deletions, and changes to configuration data. See Chapter 12 for details.
- `SolicitedNotificationByBlockingExample.cs`—Subscribes to configuration information and retrieves updates using the Blocking notification method. See Chapter 13 for details.
- `SolicitedNotificationByPollingExample.cs`—Subscribes to configuration information and retrieves updates using the Polling notification method. See Chapter 13 for details.
- `UnsolicitedNotificationExample.cs`—Subscribes to configuration information and retrieves updates using the Unsolicited Notification method. See Chapter 14 for details.

All configuration examples are located in the directories beginning with `Test`.

Example Presentation

The remaining chapters in this document focus on one example, listing the code in text format and adding commentary to help you understand the example. You can compile and run these examples to demonstrate basic SDK Service functionality and to help you plan your client application development.

Each example in the following chapters will include:

- A short introduction to the functionality.
- The example code broken into sections, with commentary.
- A stand-alone version of the example, including a sample response.

Software Requirements

All examples require GIS 7.*x* and either Configuration Server 7.*x* or CS Proxy 6.5. GIS has its own software requirements, which are described in the *Genesys Integration Server 7.6 Deployment Guide*. For a current list of supported operating systems for GIS, see *Genesys 7 Supported Operating Systems and Databases*.

Note: Start GIS and Configuration Server (or CSProxy) before running the examples.

Code stubs or proxies must be generated from the Session and Configuration WSDL files. For convenience, the code example packages include already-generated stub/proxy files.

Note: Stub/proxy files are included for your convenience. If they do not work in your environment, see "Generating Java Stub Files" on page 39 for instructions on how to generate them for Java; for Microsoft C#, see "Generating C# Proxy Files" on page 40.

Java Examples Requirements

To run the Java examples, you must meet the following system requirements:

- Either a Window or Unix platform.
- JDK 1.4 or higher to build and run the examples.
- Apache Ant 1.7.0, if you plan to use Ant.
- Ensure your `CLASSPATH` environment variable contains the path to your JDK installation and your ANT installation

C# Examples Requirements

To build and run the C# examples, you must install Microsoft .NET Framework SDK.

You can run the C# examples as a project rather than as stand-alone examples by using Microsoft Visual Studio .NET to create a solution file and importing the examples into that solution. For more assistance, see your Microsoft Visual Studio .NET documentation.

If you use Visual Studio, you do not need to compile the examples. To run them separately, be sure to compile the examples as described in “Compiling and Running the C# Examples” on page 41.

To use these examples within a Visual Studio project, you must add references to the following .NET libraries to the project:

- `System.Web.Services.dll`
- `System.XML.dll`
- `System.Web.dll`
- `System.Runtime.Remoting.dll`
- `System.Data.dll`

Development Tools

You can use any number of technologies to develop a client application for GIS. Most major market toolkits should develop successful client applications.

The following have been tested and are officially supported:

- Microsoft .NET Framework SDK v. 1.1:
  `http://msdn.microsoft.com/netframework/`
- Apache AXIS toolkit, version 1.1:
  `http://xml.apache.org/axis/index.html`

Note: You do not need to install Microsoft Visual Studio .NET to develop a Configuration SDK Service client application or to run the C# examples. However, the C# examples are constructed as a solution that you can run using Visual Studio .NET.

Installing the AXIS Toolkit

Apache Axis Toolkit is part of the Java Examples. It can be found in the `lib` directory.

If you plan to develop your application using Java, verify that your client host has the Java Development Kit (JDK) version 1.4 installed.

Java installation files can be downloaded from the following locations:

- Windows users should download the necessary software from the Sun Java site at `http://java.sun.com`.
- Solaris users should download the necessary software from the Sun Java site at `http://java.sun.com`.
- AIX users should download the necessary software from `http://www-106.ibm.com/developerworks/java/jdk/aix/`.
- HP-UX users should download the necessary software from `http://www.hp.com/products1/unix/java/`.

Installing the Microsoft .NET Framework SDK

Follow the directions on the Microsoft website to download and install Microsoft .NET Framework SDK.

If you choose to, you can use the example files to create a Visual Studio project. However, the examples run successfully using only the flat files and the .NET Framework SDK.

Note: One of the proxy generators supplied with .NET (Sproxy) is not compatible with GIS. To develop client applications in Visual Basic or C++, Genesys recommends that you use Microsoft SOAP Toolkit, version 3.0.

Using Simulators for Initial Development

To develop a client application for Genesys Integration Server (GIS), you may want to set up one or more hosts specifically dedicated to development.

If you do not want to use your live production system for developing your client application, use the Genesys sample testing tools, which include a generic T-Server, a generic PBX switch simulator, and a test simulator that simulates agent activities. These are available on the separate *Genesys Developer Program 7 Simulator Test Toolkit* CD.

Install and Compile the Examples

This section explains how to perform all the setup necessary to run the code examples for the Session and Configuration services. These code examples provide a model for development of your Genesys Integration Server (GIS) based client application.

This section does *not* include installation and configuration procedures for GIS. These are available in the *Genesys Integration Server 7.6 Deployment Guide*.

Note: All code examples are available through the DevZone Genesys Developer Documentation portal (http://www.genesyslab.com/developer), and on the Genesys Developer Documentation CD.

Procedure: Unpack the Code Examples

Purpose: To unpack the Web Services Configuration Examples.

Start of procedure

For Windows:

1. Open the appropriate `.zip` file.
2. Extract the files to a new directory, such as `C:\GCTI\Java Configuration SDK WS Examples\`.

For UNIX:

1. Copy the contents of the appropriate `tar.gz` file to a new directory.
2. Run `gunzip`.
3. Un-tar the `tar` file into a new directory, such as `GCTI/Java Configuration SDK WS Examples`.

End of procedure

Next Steps

- No further steps are required.

Example Directory Structure

The directory structure for the examples inside your newly-created directory should be as follows.

Directory Structure for Java Examples

Figure 7 shows the directory structure of the Java examples.

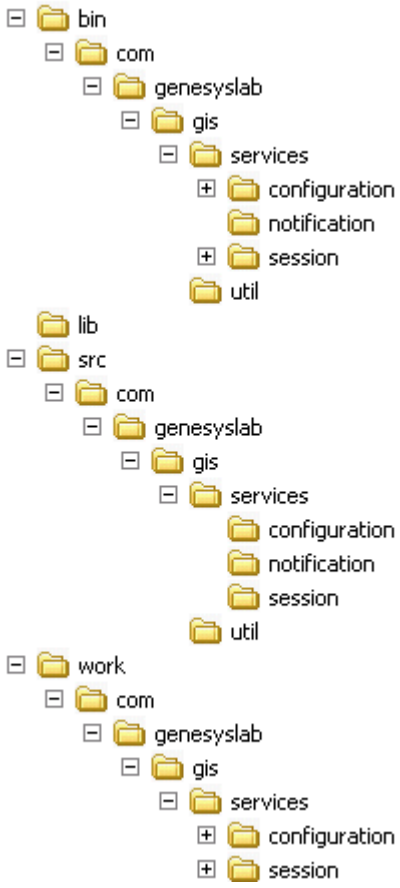

Figure 7: Web Services Java Examples Directory Structure

- `gis.properties`—This file contains the information needed to connect to GIS.
- `bin`—This directory contains all of the java class files.
- `lib`—This directory contains all the Third Party libraries (Apache Axis Toolkit, ...) used in the examples.
- `src`—This directory contains all of the source files for all examples. The directory includes the following:
  - `deploy.wsdd`—This file contains the information regarding services and defines how incoming and outgoing messages are processed.
  - com/genesyslab/gis/—This directory contains the source files for all the examples. The directory includes the following subdirectories:
    - `notification`—This directory contains the following Notification Service Java source file:
      - `NotificationService.java`
    - `session`—This directory contains the following Session Service Java source files:

- ConnectSessionServiceExample.java
- CreateSessionExample.java
- IdentifyServicesExample.java

- configuration—This directory contains the following Configuration Service Java source files:
  - GetConfigurationInformationExample.java
  - RegisterToConfigurationServiceExample.java
  - SolicitedNotificationByBlockingExample.java
  - SolicitedNotificationByPollingExample.java
  - UnsolicitedNotificationExample.java
  - UpdateConfigurationDataExample.java
- util—This directory contains the following Java files that are used to configure properties and defines how services process incoming and outgoing message:
  - NotificationModule.java
  - PropertiesLoader.java

- work—This directory contains all of the source files generated with Apache Axis Toolkit 1.4.

Directory Structure for C# Examples

Figure 8 shows the directory structure of the C# examples.

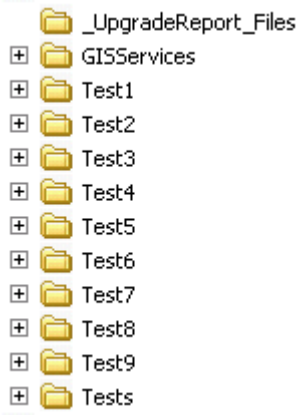

Figure 8: Web Services C# Examples Directory Structure

- Configuration SDK WS Examples.sln—This is the Visual Studio Solution file.
- GISServices—This directory contains the Visual Studio Project files for the GISServices project. The directory includes the following files that are needed to run the examples:
  - GISServices.csproj
  - SessionServiceService.cs
  - GCE.cs

  For instructions that describe how to use the files or for generating new ones, see “Generating C# Proxy Files” on page 40.

- `Test1`—This directory contains the Visual Studio Project files for the `Test1` project. The directory includes the following files needed to run the examples:
  - `Test1.csproj`
  - `CreateSessionExample.cs`
- `Test2`—This directory contains the Visual Studio Project files for the `Test2` project. The directory includes the following files needed to run the examples:
  - `Test2.csproj`
  - `ConnectSessionServiceExample.cs`
- `Test3`—This directory contains the Visual Studio Project files for the `Test3` project. It include the following files needed to run the examples:
  - `Test3.csproj`
  - `IdentifyServicesExample.cs`
- `Test4`—This directory contains the Visual Studio Project files for the `Test4` project. The directory includes the following files needed to run the examples:
  - `Test4.csproj`
  - `RegisterToConfigurationServiceExample.cs`
- `Test5`—This directory contains the Visual Studio Project files for the `Test5` project. The directory includes the following files needed to run the examples:
  - `Test5.csproj`
  - `GetConfigurationInformationExample.cs`
- `Test6`—This directory contains the Visual Studio Project files for the `Test6` project. The directory includes the following files needed to run the examples:
  - `Test6.csproj`
  - `UpdateConfigurationDataExample.cs`
- `Test7`—This directory contains the Visual Studio Project files for the `Test7` project. The directory includes the following files needed to run the examples:
  - `Test7.csproj`
  - `SolicitedNotificationByBlockingExample.cs`
- `Test8`—This directory contains the Visual Studio Project files for the `Test8` project. The directory includes the following files needed to run the examples:
  - `Test8.csproj`
  - `SolicitedNotificationByPollingExample.cs`
- `Tests`—This directory contains the Visual Studio Project files for the `Tests` project. The directory includes the following files needed to run the examples:
  - `Tests.csproj`
  - `TestSuite.cs`

Setting Your Classpath Environment Variable

The Java and Ant directories must be present in your `CLASSPATH` environment variable in order to make their executable programs available in your environment. You may find it useful to create a script, for example:

For UNIX:

```
JAVA_HOME=/<installation_directory>
ANT_HOME=/<installation_directory>/apache-ant-1.7.0
export PATH=$JAVA_HOME/bin:$ANT_HOME/bin:$PATH
```

For Windows:

```
set JAVA_HOME=C:\<installation_directory>
set ANT_HOME=C:\<installation_directory>\apache-ant-1.7.0
set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
```

Generate and Compile Stub/Proxy Files

The Configuration WSDL file defines the messages, operations, SOAP bindings, and elements that make up the GIS interface to License Manager and CSProxy. Client behavior must adhere to the formats defined in these files.

The Java stubs and classes created by WSDL2Java and the C# proxy files remove the need to work directly with the Session and Configuration WSDL files. These files function as interpreters, taking your Java or C# code and translating it into SOAP messages that GIS can use.

Note: The *Configuration SDK 7.6 Web Services API Reference* contains complete lists and explanations of all messages and operations included in the Configuration and Session WSDL files.

Generating Java Stub Files

To create a Java client application, you can use the WSDL2Java tool that is a part of the Apache AXIS Toolkit to generate a set of stub files based on the Session and Configuration WSDL files.

Using the Included Stub Files

The code example set includes a set of stub files that you can use to run the examples, located in `work\com\genesyslab\gis\services\*.java`. If you prefer—or if these files do not work in your environment—use the following procedure to generate your own stub files.

To do this using the WSDL2Java tool available from `java org.apache.axis.wsdl.WSDL2Java`:

For the Session Service, enter:

```
com.genesyslab.gis.services.session
http://<GIS_HOST>:8080/gis/services/SessionService?wsdl
```

For the Configuration Service, enter:

```
com.genesyslab.gis.services.configuration http://<GIS_HOST>:8080/
gis/services/CSProxyService?wsdl
```

Note: These examples assume that you installed GIS on port 8080. If you used a different port number, replace port 8080 to the port number you used.

The WSDL2Java tool generates all classes required for your client-side development, including type and stub classes.

Generating C# Proxy Files

The examples package includes the GIS client side proxies, `gis_service.cs` and `cfg_service.cs`, needed to run the examples. If these proxies do not match your needs or your environment, you can generate your own proxies using the `wsdl.exe` tool from the Microsoft .NET Framework SDK.

Warning! Proxies generated using the .NET utility contain declarations that are not compatible with the Configuration SDK Service `update` and `subscribeEx` methods. If you generate your own proxies, you must replace the declarations for those methods in the new files with those in the proxies provided by Genesys.

For the Session Service, enter:

```
http://<GIS_HOST>:8080/gis/services/SessionService?wsdl
```

For the Configuration Service, enter:

```
http://<GIS_HOST>:8080/gis/services/CSProxyService?wsdl
```

Note: These examples assume that you installed GIS on port 8080. If you used a different port number, replace port 8080 to the port number you used.

For further guidance in using the Microsoft .NET Framework SDK, see the user guide provided with it.

Configuring the Examples for Your Environment

Every sample file contains configurable parameters which are specific to each location. These parameters *must* be changed before executing the projects. They include connection parameters to the Session and Configuration Services (at the beginning of every source file of every project) and some specific options.

- Connect And Retrieve Configuration Data Example—Before running, change the names of the real objects in the XPath expressions.

- Update Configuration Data Example—Before running, change the names of the XML data files to be uploaded into the Configuration Service.

  This example uses two XML data files, `sample_01.xml` and `sample_01x.xml`.

Compiling the Code Examples

Whether you are using Java or C#, you must compile the examples before you can run them.

Note: You can run the C# examples as a project in Microsoft Visual Studio .NET. If you do so, you do not need to compile the examples.

To use these examples within a project, you must add references to the .NET `System.Web.Services.dll`, `System.XML.dll`, and `System.Web.dll` to the project.

Compiling the Java Examples

To compile the Java examples, use following command:

```
C:\GCTI\Java Configuration SDK WS Examples\ant compile
```

Compiling and Running the C# Examples

To run the C# examples, open the `Tests` project in Microsoft Visual Studio by clicking on `Tests.csproj`. Open `TestSuite.cs and` set the GIS host and port in each of the `Test` methods. Use the example below as a guide:

```
public void executeTest1()
{
   CreateSessionExample example = new CreateSessionExample();
   example.execute("<gis_host>:<gis_port>");
}
```

Once the `TestSuite.cs` has been modified and saved, rebuild the examples:

1. Open the Visual Studio Solution File, `Configuration SDK WS Examples.sln`.
2. Right-click on the solution tree and choose `Rebuild Solution`.

You can now run the examples by selecting, `Debug/Start Without Debugging` from the Microsoft Visual Studio menu. The examples will be executed one at a time.

Chapter

3 Getting Started—the Java Session Service Examples

The Session Service is a GIS API used by the Statistics SDK, Configuration SDK, and Interaction SDK Services. You use the Session Service to log in and log out, to generate unique Session IDs, and to verify licenses. You do not need a separate license to use the Session Service.

This chapter includes the following sections:

Session Examples Overview

Three Session examples are provided to demonstrate the actions you can take using the Session Service:

- Create Session—The first example shows how to access and log into a GIS service, and how to obtain a unique Session ID. You will need to include this Session ID in subsequent session and configuration request messages.
- Connect Session Service—This example demonstrates how to check out a license for a GIS service. This service information is passed through the constructor from any calling class that uses the `ConnectSessionServiceExample` object.
- Identify Services—Building on the previous examples, this example demonstrates how to browse the services available, and how to release checked-out services.

Create Session Example

The Create Session example code discussed below is located in the `src\com\genesyslab\gis\services\session\CreateSessionExample.java` file. The methods included in this example are described in Table 1.

Table 1: Create Session Methods

Method Name	Method Description
`loginAndCreateSession()`	This method contains the main logic of the example. This method uses the `SessionServiceServiceLocator` stub class (generated by the WSDL2Java utility) to communicate with the Session Service.
`main()`	A `main()` method is also present, allowing you to run this class as a stand-alone program.

Note: This example relies on previously generated stubs to make SOAP requests and responses. See the `work` folder for the stub classes.

Create Session Example Code

The Create Session example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.session;
import com.genesyslab.gis.services.session.types.Identity;
import com.genesyslab.gis.util.PropertiesLoader;

public class CreateSessionExample {
   public CreateSessionExample() {}
   public SessionService port;
   public String sid;
```

loginAndCreateSession() Method

This method logs in to the GIS server (using host and port information specified in the `gisServer` parameter) and retrieves a unique Session ID. The Session ID is stored in the `sid` string so that it can be reused as long as the `CreateSessionExample` object remains in scope.

```
   public void loginAndCreateSession(String gisServer) throws Exception {
```

```
    String url = "";
    String url_sid = "";
    SessionServiceServiceLocator service = new SessionServiceServiceLocator();
    port = service.getSessionService();
```

You can retrieve a `SessionService` object, represented by the port variable, by using the `getSessionService()` method of the `SessionServiceServiceLocator`.

```
    url = "http://" + gisServer + "/gis/services/SessionService";
    ((SessionServiceSoapBindingStub) port)._setProperty(
       javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY, url);
```

You must specify a URL that points to the GIS session service and the SOAP binding. The code above uses the session service SOAP binding stub `_setProperty()` method to accomplish this.

```
    Identity identity = new Identity();
    identity.setPrincipal("default");
    identity.setCredentials("password");
    identity.setTenant("Resources");
    System.out.println("logging into session server...");
    sid = port.login(identity);
    url_sid = url + "?GISsessionId=" + sid;
    System.out.println("logged into session server successfully. Session id =" + sid);
```

Log into the GIS Server by configuring an `Identity` object and passing it to the `login` request. If the `login` request is successful, then the GIS server will return a Session ID that can be used by clients to make service requests.

```
    //Bind session address endpoint with session id
    ((SessionServiceSoapBindingStub)port)._setProperty(
       javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY, url_sid);
}
```

Finally, the code calls the SOAP binding stub `_setProperty()` method again, this time passing in the Session ID as an argument. All future requests made through the `SessionService` object and containing this Session ID pertain to this particular session.

main() Method

This method creates a new instance of the `CreateSessionExample` object, uses the `loginAndCreateSession()` method by passing in the GIS server host name and port number as a `String` argument, and then logs out of the session.

```
public static void main(String[] args) {
    String gisHost = PropertiesLoader.getOption("gis.host");
```

```
      String gisPort = PropertiesLoader.getOption("gis.port");
      try {
         CreateSessionExample example = new CreateSessionExample();
         example.loginAndCreateSession(gisHost + ":" + gisPort);
         System.out.println("logging out of session server...");
         example.port.logout(example.sid); //logout request
         System.out.println("completed. Client logged out of session server...");
      } catch (Exception serviceException) {
         System.out.println(serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running the Create Session example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Create Session Example"
Buildfile: build.xml

Create Session Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221033100325H1
     [java] logging out of session server...
     [java] completed. Client logged out of session server...
```

Connect Session Service Example

The Connect Session Service example code discussed below is located in the `src\com\genesyslab\gis\services\session\ConnectSessionServiceExample.java` file. The methods included in this example are described in Table 2.

Table 2: Connect Session Service Methods

Method Name	Method Description
`execute()`	This method creates a session, checks out a license, and then prints the services license that was retrieved.
`main()`	This is a stand-alone option that allows you to test this example. The `main()` method creates a new instance of this object and starts the process.

This example uses code from the Create Session example to log into the GIS server and establish a session.

Connect Session Service Example Code

The Connect Session Service example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.session;
import com.genesyslab.gis.util.PropertiesLoader;

public class ConnectSessionServiceExample {
   public SessionService port;
   public String sid;
   public String[] services;
   public String gisServer; //keeps track of this info for other classes
   public ConnectSessionServiceExample(String[] gisServices){
      services = gisServices;
   }
```

execute() Method

This method uses a `CreateSessionExample` object to log into a GIS server, retrieve a Session ID, and check out a license for a GIS service.

```
   public void execute(String gisServer)throws Exception{
      CreateSessionExample sess = new CreateSessionExample();
      sess.loginAndCreateSession(gisServer);
      port = sess.port;
      sid = sess.sid;
      this.gisServer = gisServer;
```

First, the method creates a new instance of the `CreateSessionExample` object to log into GIS server and retrieve a Session ID.

```
      String[] list_subscribed = port.getServices(services);
```

Then it sends a request to check out the GIS Configuration Service using the `port.getServices()` method. The variable `list_subscribed` contains the list of services checked out. For the configuration examples, the checked out service will be `GIS_CONFIGSERVICE`.

```
      System.out.println("Number of services checked out = "+
         list_subscribed.length);
      for(int i=0;i<list_subscribed.length;i++){
         System.out.println("Service checked out = " + list_subscribed[i]);
```

```
      }
   }
```

Finally, the number of services checked out is printed, along with the name of license retrieved.

main() Method

The parameter `args` must contain a string array with the services you want to check out. Valid services include `GIS_STATSERVICE` or `GIS_CONFIGSERVICE`.

For example, you could enter:
`java gis.SessExamples.ConnectSessionService GIS_CONFIGSERVICE`

```
   public static void main(String[] args) {
      String gisHost = PropertiesLoader.getOption("gis.host");
      String gisPort = PropertiesLoader.getOption("gis.port");
      try{
         ConnectSessionServiceExample cs = new ConnectSessionServiceExample(args);
         cs.execute(gisHost + ":" + gisPort);
      } catch (Exception serviceException) {
         System.out.println(serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running the Connect Session Service example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Connect Session Service Example"
Buildfile: build.xml

Connect Session Service Example:
   [java] logging into session server...
   [java] logged into session server successfully.
      Session id =SOAP:SessionService:1221033247043H2
   [java] Number of services checked out = 1
   [java] Service checked out = GIS_CONFIGSERVICE
```

Identify Services Example

The Identify Services example code discussed below is located in the `src\com\genesyslab\gis\services\session\IdentifyServicesExample.java` file. The methods included in this example are described in Table 3.

Table 3: Identify Services Methods

Method Name	Method Description
`execute()`	This method creates a session, checks out a license, and then prints the services license that was retrieved.
`main()`	This is a stand-alone option that allows you to test this example. The `main()` method creates a new instance of this object and starts the process.

This example uses code from the Connect Session Service example to create a session and check out a license for the requested service.

Note: To browse and identify services, you must already have a service checked out.

Identify Services Example Code

```
package com.genesyslab.gis.services.session;

import com.genesyslab.gis.util.PropertiesLoader;

public class IdentifyServicesExample {
   public String[] services;
   public IdentifyServicesExample(String[] gisServices) {
   services = gisServices;
}
```

execute() Method

This method uses the Connect Session Service example to check out a license for the requested service. It then browses and releases the licenses that are checked out, before finally logging out of the session.

```
public void execute(String gisServer) throws Exception {
   ConnectSessionServiceExample cs = new ConnectSessionServiceExample(services);
   cs.execute(gisServer);
```

The `ConnectSessionServiceExample` object log into the GIS server and subscribes to a service, as described on page 47.

```
   String[] licenses_checked = cs.port.browseServices();
```

```
   for(int i=0;i<licenses_checked.length;i++)
      System.out.println("browsing services available: " + licenses_checked[i]);
```

Using the `ConnectSessionServiceExample` object's exposed port, a list of available services is identified and then printed.

```
   System.out.println("releasing services...");
   String[] licenses_released = cs.port.releaseServices(services);
   for(int i=0;i<licenses_released.length;i++)
      System.out.println("services released: " + licenses_released[0]);
   cs.port.logout(cs.sid); //logout request
   System.out.println("logout from session");
}
```

Using the `ConnectSessionServiceExample` object's exposed port, all checked out services are released. A list of released services is then printed and a logout request is sent to close the session.

main() Method

This is the stand-alone option to test this example. It creates a new instance of this object and calls the `execute` method.

```
public static void main(String[] args) {
   String gisHost = PropertiesLoader.getOption("gis.host");
   String gisPort = PropertiesLoader.getOption("gis.port")
   try {
      IdentifyServicesExample example = new IdentifyServicesExample(args);
      example.execute(gisHost + ":" + gisPort);
   } catch (Exception serviceException) {
      System.out.println("Exception occured" + serviceException.getMessage());
   }
}
```

Sample Output

The following sample output shows one possible result when running the Identify Services example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Identify Services Example"
Buildfile: build.xml

Identify Services Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221033348449H3
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
```

```
[java] browsing services available: GIS_CONFIGSERVICE
[java] releasing services...
[java] services released: GIS_CONFIGSERVICE
[java] logout from session
```

Chapter

4 Register to Configuration Service Example—Java

This chapter includes the following sections:

Example Overview

This example demonstrates how to register for a Configuration service, as well as how to release the service and log out from GIS.

Register to Configuration Service Example

The Register to Configuration Service example code is located in the `src\com\genesyslab\gis\services\configuration\RegisterToConfigurationServiceExample.java` file. The methods included in this example are described in Table 4 on page 54.

Table 4: Register to Configuration Service Methods

Method Name	Method Description
`register()`	This method uses a Session Example `ConnectSessionServiceExample` object to log into GIS and retrieve a session ID. Then it registers with a Configuration service by logging in.
releaseAndLogout()	This method releases the Configuration service and logs out of the current session.
`main()`	Use this method to run this class as a stand-alone program.

Register to Configuration Service Example Code

The Register to Configuration Service example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;
import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.util.PropertiesLoader;

public class RegisterToConfigurationServiceExample{
   public ConnectSessionServiceExample cs;
   public GCESoapPort configPort;
   public GCELocator configService = new GCELocator();
   String[] services = new String[]{"GIS_CONFIGSERVICE"};
```

register() Method

This method requires GIS host and port information to specified in the `gisServer` parameter, using the following format: `<gisHost>:<gisPort>`.

```
   public void register(String gisServer) throws Exception {
      cs = new ConnectSessionServiceExample(services);
      cs.execute(gisServer);
```

Establish a GIS session using the `ConnectSessionServiceExample` object.

```
      String cfgUserName = "default";
      String cfgUserPassw = "password";
      String cfgAppName = "default";
```

Specify the connection parameters you will use to connect to and register with GIS and the Configuration service.

```
      configService.setMaintainSession(true);
      configPort = configService.getGCESoapPort(new java.net.URL("http://" +
         gisServer + "/gis/services/CSProxyService?GISsessionId=" + cs.sid));
      configPort.register(cfgUserName,cfgUserPassw,cfgAppName);
      System.out.println("done");
   }
```

Create the Configuration service by using the session ID received earlier, and then register with the Configuration Server.

releaseAndLogout() Method

This method releases the service and logs out of the GIS session.

```
   public void releaseAndLogout() throws Exception {
      System.out.print("Releasing the configuration service license...");
      cs.port.releaseServices(services);
      System.out.println("done");
      System.out.println("Logging out of the GIS session ...");
      cs.port.logout(cs.sid);
      System.out.println("Logged out successfully.");
   }
```

main() Method

This is the stand-alone option you can use to test this example. It creates a new instance of this object and starts the register and release processes.

```
   public static void main(String[] args) {
      String gisHost = PropertiesLoader.getOption("gis.host");
      String gisPort = PropertiesLoader.getOption("gis.port");
      try{
         RegisterToConfigurationServiceExample sb = new
            RegisterToConfigurationServiceExample();
         sb.register(gisHost + ":" + gisPort);
         sb.releaseAndLogout();
      } catch (Exception serviceException) {
         System.out.println(serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running the Register To Configuration Service example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Register To Configuration Service"
Buildfile: build.xml

Register To Configuration Service Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221138459464H1
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
     [java] done
     [java] Releasing the configuration service license...done
     [java] Logging out of the GIS session ...
     [java] Logged out successfully.
```

Chapter

5 Get Configuration Information Example—Java

This chapter includes the following sections:

Example Overview

This example demonstrates how you use query expressions to retrieve Configuration information from an established Configuration session using `get` requests, and how to process the results of your queries.

Some common query expressions are included as examples in the code. These queries are listed and described in "Query Analysis" on page 61.

Note: All Genesys SDKs use a proprietary query language that is based on a subset of the XPath expression language. Standard XPath queries will usually work, however, not all XPath functionality is supported.

For more information on how to construct queries, see the *Configuration SDK 7.6 Web Services API Reference*.

Get Configuration Information Example

The Get Configuration Information example code discussed is located in the `src\com\genesyslab\gis\services\configuration\GetConfigurationInformati`

onExample.java file. The methods included in this example are described in Table 5.

Table 5: Get Configuration Information Methods

Method Name	Method Description
execute()	This method creates a session, logs into a GIS server, and subscribes to the licensed services. The get() method is used to send queries and return configuration data in XML format.
displayResults()	This method displays the query and results that are used as parameters. A query returns MessageElement objects reflecting the update schema structure.
main()	Use this method to run this class as a stand-alone program.

This sample explains how to accomplish Configuration service get requests. It gives examples of different query expressions that can be used, and how to process those queries.

Get Configuration Information Example Code

The Get Configuration Information example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;
import org.apache.axis.message.MessageElement;
import com.genesyslab.gis.services.configuration.types.CSResult;
import com.genesyslab.gis.util.PropertiesLoader;

public class GetConfigurationInformationExample {
   public GetConfigurationInformationExample(){}
   String queryAllAgents = "CfgAgent";
   String queryAgentGroupsAlt = "CfgAgent[@name='Lucent_1000']/agentGroups/*/baseDBID";
   String queryAgentGroups = "CfgAgentGroup[agentDBIDs/*/linkDBID/@name='Agent0']";
   String queryAgents = "CfgAgentGroup[@name='Agent Group 1']/agentDBIDs/*/linkDBID";
   String queryAgentsAlt = "CfgAgent[agentGroups/*/baseDBID/@name='Agent Group 1']";
   String querySkills = "CfgAgent[@name='Lucent_1000']/skillLevels/*/linkDBID";
   String querySwitches = "CfgAgent[@name='Lucent_1000']/agentLogins/*/
      linkDBID/ownerDBID";
   String queryAgentsByPlace = "CfgAgent[placeDBID/@name='Place_1000']";
   String queryDNs = "CfgSwitch[@name='LucentG3']/DNs/CfgDN[@type='2']";
   String queryAgentsByFolder= "CfgFolder[@name='SubFolder']/objectIDs/
```

```
   CfgPersonShortcut/linkDBID";
```

This section defines a number of queries that can be used to retrieve Configuration information. The queries are based on the CSProxy schema definition, and are described further in "Query Analysis" on page 61.

execute() Method

This method creates a session, logs into a GIS server, and subscribes to the licensed services. The `get()` method is used to retrieve query results, which consist of configuration data in XML format.

```
public void execute(String targetHost) throws Exception{
   RegisterToConfigurationServiceExample subConfig = new
      RegisterToConfigurationServiceExample();
   subConfig.register(targetHost);//register to GIS configuration service
   System.out.println("==>Configuration service version is ..." +
      subConfig.configPort.getVersion());
```

Once the session is established and the Configuration service is registered, the version number is displayed.

```
   displayResults(queryAllAgents,subConfig.configPort.get(queryAllAgents));
   displayResults(queryAgentGroups,subConfig.configPort.get(queryAgentGroups));
   displayResults(queryAgents,subConfig.configPort.get(queryAgents));
   displayResults(querySkills,subConfig.configPort.get(querySkills));
   displayResults(querySwitches,subConfig.configPort.get(querySwitches));
   displayResults(queryAgentsByPlace,subConfig.configPort.get(queryAgentsByPlace));
   displayResults(queryDNs,subConfig.configPort.get(queryDNs));
   displayResults(queryAgentsByFolder,subConfig.configPort.get(queryAgentsByFolder));
   System.out.println("\n");
```

Each query is sent to the Configuration service using the `get` request, and both the query and results are then displayed using the `displayResults()` method.

```
   subConfig.releaseAndLogout();
}
```

The Configuration services are released, and the session is closed before exiting the `execute()` method.

displayResults() method

This method displays the query and results that are used as parameters. A query returns `MessageElement` objects reflecting the `update` schema structure.

```
void displayResults(String query,CSResult result) {
```

```
      if (result.get_any()[0].getChildren() != null) {
         System.out.println("\n");
         //writes a newline char. Makes it easier to read the console output.
         System.out.println("Result for query '" + query + "'");
         for (int i=0; i < result.get_any()[0].getChildren().size(); i++) {
            MessageElement elem =
               (MessageElement)result.get_any()[0].getChildren().get(i);
            System.out.println(" =>Object type is = '" + elem.getName() +
               "' Object name is = '" + elem.getAttributeValue("name") + "'");
         }
      } else{
         System.out.println("\n");
         //writes a newline char. Makes it easier to read the console output.
         System.out.println("Results for query '" + query + "'" );
         System.out.println(" => No data available");
      }
   }
```

main() Method

This is a stand-alone option to test this example. The `main()` method creates a new instance of this object and starts the `execute` process.

```
   public static void main (String args[]) throws Exception {
      String gisHost = PropertiesLoader.getOption("gis.host");
      String gisPort = PropertiesLoader.getOption("gis.port");
      GetConfigurationInformationExample configInfo = new
         GetConfigurationInformationExample();
      configInfo.execute(gisHost + ":" + gisPort);
   }
}
```

Sample Output

The following sample output shows one possible result when running the Get Configuration Information Example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Get Configuration Information Example"
Buildfile: build.xml

Get Configuration Information Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221138682926H2
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
     [java] done
     [java] ==>Configuration service version is ...7.5.000.11
     [java]
```

```
[java]
[java] Result for query 'CfgAgent'
[java]  =>Object type is = 'CfgAgent' Object name is = 'Meridian_8000'
[java]
.....
[java]
[java] Results for query 'CfgAgentGroup[agentDBIDs/*/linkDBID/@name='Agent0']'
[java]  => No data available
[java]
[java]
[java] Results for query 'CfgAgentGroup[@name='Agent Group']/agentDBIDs/
   */linkDBID'
[java]  => No data available
[java]
[java]
[java] Result for query 'CfgAgent[@name='Lucent_1000']/skillLevels/*/linkDBID'
[java]  =>Object type is = 'CfgSkill' Object name is = 'StupidAgent'
[java]
[java]
[java] Result for query 'CfgAgent[@name='Lucent_1000']/agentLogins/*/
   linkDBID/ownerDBID'
[java]  =>Object type is = 'CfgSwitch' Object name is = 'LucentG3'
[java]
[java]
[java] Result for query 'CfgAgent[placeDBID/@name='Place_1000']'
[java]  =>Object type is = 'CfgAgent' Object name is = 'Lucent_1000'
[java]
[java]
[java] Results for query 'CfgSwitch[@name='LucentG3']/DNs/CfgDN[@type='2']'
[java]  => No data available
[java]
[java]
[java] Results for query 'CfgFolder[@name='SubFolder']/objectIDs
   /CfgPersonShortcut/linkDBID'
[java]  => No data available
[java]
[java]
[java] Releasing the configuration service license...done
[java] Logging out of the GIS session ...
[java] Logged out successfully.
```

Query Analysis

The queries used to retrieve Configuration information included in this example are described in Table 6 on page 62. Queries are based on the CSProxy schema definition and use a Genesys proprietary query language that is based on a subset of the XPath query format.

Table 6: List of Queries

Query Description	Query Name	Query
Retrieve all available Agents.	queryAllAgents	CfgAgent
Retrieve all Agent Groups to which a specific agent belongs.	• queryAgentGroups • queryAgentGroupsAlt	• CfgAgentGroup[agentDBIDs/*/linkDBID/@name='<AgentID>'] • CfgAgent[@name='<AgentID>']/agentGroups/*/baseDBID
Retrieve all Agents from a specific Agent Group.	• queryAgents • queryAgentsAlt	• CfgAgentGroup[@name='<AgentGroup>']/agentDBIDs/*/linkDBID • CfgAgent[agentGroups/*/baseDBID/@name='<AgentGroup>']
Retrieve all Skills for a specific Agent.	querySkills	CfgAgent[@name='<AgentID>']/skillLevels/*/linkDBID
Retrieve all Switches connected to a specific Agent through the agent Login.	querySwitches	CfgAgent[@name='<AgentID>']/agentLogins/*/linkDBID/ownerDBID
Retrieve all Agents assigned to a particular Place.	queryAgentsByPlace	CfgAgent[placeDBID/@name='<PlaceName>']
Retrieve all Places with no assigned Agents.	queryPlacesWoAgents	CfgPlace[not agents]
Retrieve all DNs of the type ACD Position that belong to a specific Switch.	queryDNs	CfgSwitch[@name='<SwitchName>']/DNs/CfgDN[@type='<DnType>']
Retrieve all Agents in a specific Folder.	queryAgentsByFolder	CfgFolder[@name='<FolderName>']/objectIDs/CfgPersonShortcut/linkDBID

Chapter

6 Update Configuration Data Example—Java

This chapter includes the following sections:

Example Overview

This example uses `update` requests to modify Configuration service information using by uploading an XML data file, and shows how to process the results of the request. The actual sequence of updates is contained in a separate XML data file (`\src\sample_01.xml`) for readability.

Update Configuration Data Example

The Update Configuration Data example code is located in the `src\com\genesyslab\gis\services\configuration\UpdateConfigurationDataExample.java` file. The methods included in this example are described in Table 7 on page 64.

Table 7: Update Configuration Data Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then reads and prepares the XML update sequence, sends an `update` request to the Configuration Server, and displays the response.
`main()`	Use this method to run this class as a stand-alone program.

Update Configuration Data Example Code

The Update Configuration Data example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;
import org.apache.axis.message.MessageElement;
import com.genesyslab.gis.services.configuration.types.CSResult;
import com.genesyslab.gis.services.configuration.types.CSXmlData;
import com.genesyslab.gis.util.PropertiesLoader;

public class UpdateConfigurationDataExample {

   String gisPrincipal = "default";
   String gisCredentials = "password";
   String cfgUserName = "default";
   String cfgUserPassw = "password";
   String cfgAppName = "default";
   static String sampleFile = "sample_01.xml"; //default
   String dataFile = sampleFile;
```

Specify the parameters used to connect to GIS and register with the Configuration service, along with the location of the XML data file holding the update sequence.

execute() Method

This method requires GIS host and port information to specified in the `gisServer` parameter, using the following format: `<gisHost>:<gisPort>`.

The `execute` method accomplishes all the application logic for the example.

```
public void execute(String gisServer) throws Exception {
   RegisterToConfigurationServiceExample subConfig = new
      RegisterToConfigurationServiceExample();
   subConfig.register(gisServer); //subscribe to GIS configuration service
```

The `RegisterToConfiguration` class is used to establish a session and subscribe to the GIS Configuration service.

```
   javax.xml.parsers.DocumentBuilderFactory documentBuilderFactory =
      javax.xml.parsers.DocumentBuilderFactory.newInstance();
   javax.xml.parsers.DocumentBuilder documentBuilder =
      documentBuilderFactory.newDocumentBuilder();
   org.w3c.dom.Document xmlDocument = documentBuilder.parse(this.getClass()
      .getClassLoader().getResourceAsStream(dataFile));
   CSXmlData updateData = new CSXmlData();
   MessageElement messageElement = new
      MessageElement(xmlDocument.getDocumentElement());
   MessageElement[] any = new org.apache.axis.message.MessageElement[]
      {MessageElement};
   updateData.set_any(any);
```

Parse and prepare the XML data file, so that the data can be sent to the Configuration service using an `update` request.

```
   System.out.print("==>Updating configuration using " + dataFile + " ...");
   CSResult updateResponse = subConfig.configPort.update(updateData);
```

Use the Configuration service `update` request, and store the results in the `updateResponse` object.

```
   System.out.println("done \n");
   System.out.println("==>Update result:");
   for (int j=0; j < updateResponse.get_any()[0].getChildren().size(); j++) {
   MessageElement elem =
      (MessageElement)updateResponse.get_any()[0].getChildren().get(j);
   MessageElement errorInfo = (MessageElement)elem.getChildElements().next();
   if (errorInfo.getAttributeValue("returnCode").equals("0"))
      System.out.println(" Item : id [" + elem.getAttributeValue("id") +
         "] processed successfully");
   else{
      MessageElement serverErrorInfo =
         (MessageElement)errorInfo.getChildElements().next();
      MessageElement description =
         (MessageElement)serverErrorInfo.getChildElements().next();
      System.out.println(" Item : id [" + elem.getAttributeValue("id") +
         "] processed with error : " + description.getValue());
      }
   }
   System.out.println("\n");
```

Print the results of the update attempt.

```
   subConfig.releaseAndLogout();
}
```

The Configuration services are released, and the session is closed before exiting the `execute()` method.

main() Method

This is the stand-alone option you can use to test this example. It creates a new instance of the `UpdateConfigurationData` object and calls the `execute()` method.

```
public static void main (String args[]) throws Exception {
   String gisHost = PropertiesLoader.getOption("gis.host");
   String gisPort = PropertiesLoader.getOption("gis.port");
   UpdateConfigurationDataExample example = new UpdateConfigurationDataExample();
   example.execute(gisHost + ":" + gisPort);
}
```

Sample Output

The following sample output shows one possible result when running the Update Configuration Data example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Update Configuration Data Example"
Buildfile: build.xml

Update Configuration Data Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221139210442H4
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
     [java] done
     [java] ==>Updating configuration using sample_01.xml ...done
     [java]
     [java] ==>Update result:
     [java]  Item : id [Skill-001] processed successfully
     [java]  Item : id [Environment] processed successfully
     [java]
     [java]
     [java] Releasing the configuration service license...done
     [java] Logging out of the GIS session ...
     [java] Logged out successfully.
```

Chapter

7 Solicited Notification Examples—Java

This chapter includes the following sections:

Solicited Notification Examples Overview

Two solicited notification examples are provided to demonstrate the difference between Solicited Notification by Polling and Solicited Notification by Blocking:

- Solicited Notification by Polling—This example shows how to use polling to access the Configuration service notification mechanism. When polling, you subscribe to a Configuration service and then use the `refresh()` method to check for updates.
- Solicited Notification by Blocking—This example shows how to use asynchronous calls to the `getEvent` method to access the Configuration service notification mechanism.

This example uses XML data stored in a separate file (`\src\sample_01x.xml`) for readability. The XML file contains information to create a new Skill object, allowing you to test the notification process.

Solicited Notification by Polling Example

The Solicited Notification by Polling example code is located in the `src\com\genesyslab\gis\services\configuration\SolicitedNotificationByPo`

llingExample.java file. The methods included in this example are described in Table 8.

Table 8: Solicited Notification by Polling Methods

Method Name	Method Description
execute()	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then subscribes to receive notification about Agent Skills, and polls for updates.
main()	This method runs the SolicitedNotificationByPollingExample class as a stand-alone program.
displayNotification()	This method displays the response returned after using refresh to poll for notification updates.

Solicited Notification by Polling Example Code

The Solicited Notification by Polling example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;

import java.util.Iterator;
import org.apache.axis.message.MessageElement;
import com.genesyslab.gis.services.configuration.types.CSResult;
import com.genesyslab.gis.services.configuration.types.CSXmlData;
import com.genesyslab.gis.util.PropertiesLoader;

public class SolicitedNotificationByPollingExample {
   String gisPrincipal = "default";
   String gisCredentials = "password";
   String cfgUserName = "default";
   String cfgUserPassw = "password";
   String cfgAppName = "default";
   String[] services = new String[]{"GIS_CONFIGSERVICE"};
```

These are the connection parameters used to connect to and register with GIS and the Configuration service.

execute() Method

This method requires GIS host and port information to specified in the `gisServer` parameter, using the following format: `<gisHost>:<gisPort>`.

The `execute` method accomplishes all the application logic for the example.

```
public void execute(String gisServer) throws Exception {
   String dataFile = "sample_01x.xml";
   RegisterToConfigurationServiceExample subConfig = new
      RegisterToConfigurationServiceExample();
   subConfig.register(gisServer); //register to GIS configuration service
```

The `RegisterToConfiguration` class provides functionality to create a GIS session and register with the GIS Configuration service, or to release a service and log out of the GIS session.

```
   System.out.println("==>Subscription on notifications completed ..." +
      subConfig.configPort.subscribe("CfgSkill","",""));
   subConfig.configService.getEngine().setOption(
      org.apache.axis.AxisEngine.PROP_DOMULTIREFS,Boolean.FALSE);
   CSResult result = subConfig.configPort.refresh("");
   MessageElement checkpointValue = (MessageElement)result.get_any()[0];
   System.out.println("==>Current checkpoint is [" + checkpointValue.getValue()
      + "]");
```

Now subscribe to receive notification about an agent's skill, set the engine option to prevent `multiRef` elements in the method parameters, and get the current checkpoint.

```
   javax.xml.parsers.DocumentBuilderFactory documentBuilderFactory =
      javax.xml.parsers.DocumentBuilderFactory.newInstance();
   javax.xml.parsers.DocumentBuilder documentBuilder =
      documentBuilderFactory.newDocumentBuilder();
   org.w3c.dom.Document xmlDocument = documentBuilder.parse(this.getClass().
      getClassLoader().getResourceAsStream(dataFile));
   CSXmlData updateData = new CSXmlData();
   MessageElement messageElement = new
      MessageElement(xmlDocument.getDocumentElement());
   MessageElement[] any = new
      org.apache.axis.message.MessageElement[] { messageElement };
   updateData.set_any(any);
   System.out.print("==>Updating configuration using sample_01x.xml ...");
   subConfig.configPort.update(updateData);
   System.out.println("done");
```

Read and parse the XML document. Use the `update` request to send the XML to your Configuration service and create a new skill object.

```
   result = subConfig.configPort.refresh(checkpointValue.getValue());
   checkpointValue = (MessageElement)result.get_any()[1];
   System.out.println("===>Current checkpoint is [" +
      checkpointValue.getValue() + "]");
   MessageElement cfgData = (MessageElement)result.get_any()[1];
   displayNotification(cfgData,"CfgCreate","Created");
```

After making the update, call the `refresh()` method again. Print the current checkpoint, and use the `displayNotification()` method to display the results.

```
   subConfig.configPort.unsubscribe();
   subConfig.releaseAndLogout();
}
```

Unsubscribe from the notification request, release the Configuration service and log out of the current session before exiting the `execute()` method.

displayNotification() Method

Use this method to display the notification results after using the `refresh()` method.

```
public void displayNotification(MessageElement cfgData,String elementName,
   String textStr) {
   Iterator iter = cfgData.getChildElements();
   MessageElement oper = null;
   while (iter.hasNext())
      if (((oper = (MessageElement)iter.next()) != null) &&
            oper.getName().equals(elementName)) {
         System.out.println(" ===>" + textStr + ":");
         Iterator subIter = oper.getChildElements();
         MessageElement obj = null;
         while (subIter.hasNext())
            if ((obj = (MessageElement)subIter.next()) != null)
               System.out.println("  Object [" + obj.getName() + "],
                  DBID [" + obj.getAttributeValue("DBID") +
                  ", name [" + obj.getAttributeValue("name") + "]");
         }...
```

main() Method

This is the stand-alone option you can use to test this example. It creates a new instance of the `SolicitedNotificationByPollingExample` object and calls the `execute()` method.

```
public static void main (String args[]) throws Exception {
   String gisHost = PropertiesLoader.getOption("gis.host");
   String gisPort = PropertiesLoader.getOption("gis.port");
```

```
      SolicitedNotificationByPollingExample example = new
         SolicitedNotificationByPollingExample();
      example.execute(gisHost + ":" + gisPort);
   }...
```

Sample Output

The following sample output shows one possible result when running the Update Configuration Data example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Update Configuration Data Example"
Buildfile: build.xml

Update Configuration Data Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221139210442H4
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
     [java] done
     [java] ==>Updating configuration using sample_01.xml ...done
     [java]
     [java] ==>Update result:
     [java]  Item : id [Skill-001] processed successfully
     [java]  Item : id [Environment] processed successfully
     [java]
     [java]
     [java] Releasing the configuration service license...done
     [java] Logging out of the GIS session ...
     [java] Logged out successfully.
```

Solicited Notification by Blocking Example

The Solicited Notification by Blocking example code is located in the `src\com\genesyslab\gis\services\configuration\SolicitedNotificationByBlockingExample.java` file. The methods included in this example are described in Table 9 on page 72.

Table 9: Solicited Notification by Blocking Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then subscribes to receive notification about Skills and waits for an update.
`main()`	This method runs the `SolicitedNotificationByBlockingExample` class as a stand-alone program.
`displayNotification()`	This method displays the response returned after using `refresh` to poll for notification updates.

Solicited Notification by Blocking Example Code

The Solicited Notification by Blocking example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;

import java.util.Iterator;
import org.apache.axis.message.MessageElement;
import com.genesyslab.gis.services.configuration.types.CSResult;
import com.genesyslab.gis.util.PropertiesLoader;

public class SolicitedNotificationByBlockingExample {
   String gisPrincipal = "default";
   String gisCredentials = "password";
   String cfgUserName = "default";
   String cfgUserPassw = "password";
   String cfgAppName = "default";
```

These are the connection parameters used to connect to and register with GIS and the Configuration service.

execute() Method

This method requires GIS host and port information to specified in the `gisServer` parameter, using the following format: `<gisHost>:<gisPort>`.

The `execute` method accomplishes all the application logic for the example.

```
public void execute(String gisServer) throws Exception {
   RegisterToConfigurationServiceExample subConfig = new
      RegisterToConfigurationServiceExample();
   subConfig.register(gisServer);//register to GIS configuration service
```

The `RegisterToConfiguration` class provides functionality to create a GIS session and register with the GIS Configuration service, or to release a service and log out of the GIS session.

To get updates, first subscribe for notifications on Agents.

```
   System.out.println("Subscription on notifications completed ..." +
      subConfig.configPort.subscribe("CfgAgent","",""));
   System.out.println("Waiting for notifications ...");
   CSResult result = subConfig.configPort.getEvent();
   System.out.println("Notification arrived");
```

Subscribe for notification of Agent updates, and use the `getEvent`() request to check for updates.

Note: While waiting for a response from the `getEvent()` request, no additional actions will be taken by this thread. Progress is blocked until a response is returned.

```
   displayNotification(result.get_any()[0],"CfgCreate","Created");
   displayNotification(result.get_any()[0],"CfgUpdate","Updated");
   displayNotification(result.get_any()[0],"CfgRemove","Removed");
   subConfig.releaseAndLogout();}
```

Use the `displayNotification()` method to display the results. Then release the Configuration service and log out of the current session before exiting the `execute()` method.

displayNotification() Method

Use this method to display the notification results after using the getEvent`()` method.

```
public void displayNotification(MessageElement cfgData,String elementName,
     String textStr) {
   Iterator iter = cfgData.getChildElements();
   MessageElement oper = null;
   while (iter.hasNext())
   if (((oper = (MessageElement)iter.next()) != null) &&
        oper.getName().equals(elementName)) {
      System.out.println("\n" + textStr + ":");
      Iterator subIter = oper.getChildElements();
```

```
        MessageElement obj = null;
        while (subIter.hasNext())
           if ((obj = (MessageElement)subIter.next()) != null)
              System.out.println("  Object type = '" + obj.getName() + "', DBID = " +
                 obj.getAttributeValue("DBID") + ", name = " +
                 obj.getAttributeValue("name") + "\n");}}
```

main() Method

This is the stand-alone option you can use to test this example. It creates a new instance of the `SolicitedNotificationByBlockingExample` object and calls the `execute()` method.

```
  public static void main (String args[]) throws Exception  {
     String gisHost = PropertiesLoader.getOption("gis.host");
     String gisPort = PropertiesLoader.getOption("gis.port");
     SolicitedNotificationByBlockingExample example = new
     SolicitedNotificationByBlockingExample();
     example.execute(gisHost + ":" + gisPort);
  }...
```

Sample Output

The following sample output shows one possible result when running the Solicited Notification By Blocking example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Solicited Notification By Blocking
   Example"
Buildfile: build.xml

Solicited Notification By Blocking Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221140188481H14
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_CONFIGSERVICE
     [java] done
     [java] Subscription on notifications completed ...OK
     [java] Waiting for notifications ...
     [java] Notification arrived
     [java]
     [java] Updated:
     [java]   Object type = 'CfgAgent', DBID = 141, name = ALu_1005
     [java]
     [java] Releasing the configuration service license...done
     [java] Logging out of the GIS session ...
     [java] Logged out successfully.
```

Chapter

8 Unsolicited Notification Example—Java

This chapter includes the following section:

Example Overview

The `UnsolicitedNotificationExample()` method in the Unsolicited Notification configuration example sends a request to GIS for automatic updates to a configuration object. This process requires a Web service to be deployed on listening HTTP server, because GIS can send data only to a web server. A special client is needed to retrieve a notification message from the Web service and print it to the console.

The example consists of two files:

- `NotificationService.java`—Contains the code for a Web service to be invoked by an HTTP server (such as the Apache Tomcat server) when Configuration Server issues a notification request. This service must be deployed before you exercise the example.
- `UnsolicitedNotificationExample.java`—This file contains the code that subscribes to notifications on Configuration Server. This file should be run first.

Unsolicited Notification Example

The Unsolicited Notification example code is located in the `src\com\genesyslab\gis\services\configuration\UnsolicitedNotificationEx`

ample.java file. The methods included in this example are described in Table 10 on page 76.

Table 10: Unsolicited Notification Example Methods

Method Name	Method Description
execute()	This method creates a session, and then logs in and registers with GIS and Configuration services. It then subscribes to services, registers a channel to listen for responses on, and waits for notifications to arrive.
main()	Use this method to run this class as a stand-alone program.

Unsolicited Notification Example Code

The Unsolicited Notification example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
package com.genesyslab.gis.services.configuration;
import org.apache.axis.message.MessageElement;
import com.genesyslab.gis.services.configuration.types.CSSubscriptions;
import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.util.NotificationModule;
import com.genesyslab.gis.util.PropertiesLoader;

public class UnsolicitedNotificationExample {

   String gisPrincipal = "default";
   String gisCredentials = "password";
   String cfgUserName = "default";
   String cfgUserPassw = "password";
   String cfgAppName = "default";
   String[] services = new String[] { "GIS_CONFIGSERVICE" };
   public ConnectSessionServiceExample cs;
```

execute() Method

This method requires GIS host and port information to specified in the gisServer parameter, using the following format: <gisHost>:<gisPort>.

The execute method accomplishes all the application logic for the example.

```
   void execute(String gisServer) throws Exception {
      cs = new ConnectSessionServiceExample(services);
```

```
cs.execute(gisServer);
```

Establish a GIS session using the `ConnectSessionServiceExample` object.

```
String cfgUserName = "default";
String cfgUserPassw = "password";
String cfgAppName = "default";
```

Specify the connection parameters you want to use to connect to and register with GIS and the Configuration service.

First, create a service.

```
CELocator configService = new GCELocator();
configService.setMaintainSession(true);
GCESoapPort configPort = configService.getGCESoapPort(new java.net.URL
   ("http://" + gisServer+ "/gis/services/CSProxyService?GISsessionId="
   + cs.sid));
```

Create the Configuration service using the session ID received earlier, and register with the Configuration Server. Next, use the service to get a stub which implements the Session ID.

```
configService.getEngine().setOption(org.apache.axis.AxisEngine.PROP_DOMULTIREFS,
   Boolean.FALSE);
```

Set the engine option which prevents having 'multiRef' elements put into the method parameters.

```
configPort.register(cfgUserName, cfgUserPassw, cfgAppName);
System.out.println("done");
```

Register at the Configuration Service.

```
System.out.println("==>Configuration Service version is ..." +
   configPort.getVersion());
```

Display the Configuration Service version.

```
NotificationModule.getInstance().start();
```

Start the Notification Service used to retrieve notification Events.

```
javax.xml.parsers.DocumentBuilderFactory documentBuilderFactory =
   javax.xml.parsers.DocumentBuilderFactory.newInstance();
javax.xml.parsers.DocumentBuilder documentBuilder =
   documentBuilderFactory.newDocumentBuilder();
org.w3c.dom.Document xmlDocument = documentBuilder.newDocument();
```

```
      org.w3c.dom.Element subscriptionData =
         xmlDocument.createElement("subscriptionData");
      org.w3c.dom.Element subscription = xmlDocument.createElement("subscription");
      subscription.setAttribute("type", "CfgSkill");
      subscriptionData.appendChild(subscription);
      xmlDocument.appendChild(subscriptionData);
```

Read and parse the XML document.

```
      CSSubscriptions subscriptions = new CSSubscriptions();
      MessageElement messageElement =
         new MessageElement(xmlDocument.getDocumentElement());
      MessageElement[] any = new org.apache.axis.message.MessageElement[]
         { messageElement };
      subscriptions.set_any(any);
      String gisHost = PropertiesLoader.getOption("gis.host");
      String notificationPort = PropertiesLoader.getOption("client.notification.port");
      System.out.println("==>Subscription on notifications completed ..."
         + configPort.subscribeEx(subscriptions, "http://" + gisHost+ ":"
         + notificationPort + "/axis/services/notification", ""));
```

Now subscribe to receive notification about an agent's skill.

```
      Thread.currentThread().run();
}
```

main() Method

Use this stand-alone option to test this example. This stand-alone methods creates a new instance of the `UnpdatesolicitedNotification` object and calls the `execute()` method.

```
   public static void main(String args[]) throws Exception {
      String gisHost = PropertiesLoader.getOption("gis.host");
      String gisPort = PropertiesLoader.getOption("gis.port");
      UnsolicitedNotificationExample test = new UnsolicitedNotificationExample();
      test.execute(gisHost + ":" + gisPort);
   }
```

Sample Output

The following sample output shows one possible result when running the Unsolicited Notification example:

```
C:\GCTI\Java Configuration SDK WS Examples\ant "Unsolicited Notification Example"
Buildfile: build.xml

Unsolicited Notification Example:
```

```
[java] logging into session server...
[java] logged into session server successfully.
   Session id =SOAP:SessionService:1221140588608H16
[java] Number of services checked out = 1
[java] Service checked out = GIS_CONFIGSERVICE
[java] done
[java] ==>Configuration Service version is ...7.5.000.11
[java] - starting up SimpleAxisServer on port 50123
   (C:GCTI\Configuration SDK WS Examples)
[java] ==>Subscription on notifications completed ...OK
[java] Hello from Config notification service
[java] host 2025 port FRBRESV158668 serverapp confserv
[java] Got message...
[java] Message: <CfgData><CfgUpdate><CfgSkill name="Skill-001x"
   state="2" ownerDBID="101" tenantDBID="101" DBID="145"/></CfgUpdate></CfgData>
```

Chapter

9 Getting Started—the C# Session Examples

The Session Service is a GIS API used by the Statistics SDK, Configuration SDK, and Interaction SDK Services. You use the Session Service to log in and log out, to generate unique Session IDs, and to verify licenses. You do not need a separate license to use the Session Service.

This chapter includes the following sections:

Session Examples Overview

Three Session examples are provided to demonstrate the actions you can take using the Session Service:

- Create Session—The first example shows how to access and log into a GIS service, and how to obtain a unique Session ID. You will need to include this Session ID in subsequent session and configuration request messages.
- Connect Session Service—This example demonstrates how to check out a license for a GIS service. This service information is passed through the constructor from any calling class that uses the `ConnectSessionServiceExample` object.
- Identify Services—Building on the previous examples, this example demonstrates how to browse the services available, and how to release checked-out services.

To run these examples successfully, keep in mind that:

- Only the example file can have a `main()` method. You must comment out any `main()` methods in supporting files before running the example.
- The `unregister()` method includes the `unsubscribe()` functionality; hence, the example only uses `unregister()` in the `closeSession()` method.
- To use these examples within a C# project, you must add references to the following .NET libraries to your project:
  - `System.Web.Services.dll`
  - `System.XML.dll`
  - `System.Web.dll`
  - `System.Runtime.Remoting.dll`
  - `System.Data.dll`

Create Session Example

The Create Session example code discussed below is located in the `\Test1\CreateSessionExample.cs` file. The methods included in this example are described in Table 11.

Table 11: Create Session Methods

Method Name	Method Description
`loginAndCreateSession()`	This method contains the main logic of the example. This method uses the `SessionServiceServiceLocator` stub class (generated by the WSDL2Java utility) to communicate with the Session Service.

Create Session Example Code

The Create Session example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using GISServices;

   public class CreateSessionExample
   {
      public SessionServiceService port;
      public String sid;
```

loginAndCreateSession() Method

This method logs in to the GIS server (using host and port information specified in the `gisServer` parameter) and retrieves a unique Session ID. The Session ID is stored in the `sid` string so that it can be reused as long as the `CreateSessionExample` object remains in scope.

```
public void loginAndCreateSession(String targetHost)
{
   String url = "";
   String url_sid = "";
   port = new SessionServiceService();
   port.Url = "http://"+targetHost+"/gis/services/SessionService";
```

You must specify a URL that points to the GIS session service.

```
   Identity identity = new Identity();
   identity.principal = "default";
   identity.credentials = "password";
   identity.tenant = "Resources";
   System.Console.WriteLine("logging into session server...");
   sid = port.login(identity);
   url_sid = url + "?GISsessionId="+sid;
   System.Console.WriteLine("logged into session server successfully.
      Session id ="+sid);
   port.Url = "http://" + targetHost +
      "/gis/services/SessionService?GISsessionId=" + sid;
}
```

Log into the GIS Server by configuring an `Identity` object and passing it to the `login` request. If the `login` request is successful, then the GIS server returns a Session ID that can be used by clients to make service requests.

Sample Output

The following sample output shows one possible result when running the Create Session example:

```
Test 1
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221140940938H2
logging out of session server...
completed. Client logged out of session server...
```

Connect Session Service Example

The Connect Session Service example code discussed below is located in the `\Test2\ConnectSessionServiceExample.cs` file. The methods included in this example are described in Table 12.

Table 12: Connect Session Service Methods

Method Name	Method Description
`execute()`	This method creates a session, subscribes to the Configuration service, and then prints a list of services with subscriptions.

This example uses code from the Create Session example to log into the GIS server and establish a session.

Connect Session Service Example Code

The Connect Session Service example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.Text;
using GISServices;

   class ConnectSessionServiceExample
   {
      public SessionServiceService port;
      public String sid;
      public String[] services = new String[1]{"GIS_CONFIGSERVICE"};
```

execute() Method

This method uses `CreateSessionExample` to log onto GIS server and retrieve a session. Then it requests a subscription to the GIS Configuration service.

```
      public void execute(String targetHost)
      {
         CreateSessionExample sess = new CreateSessionExample();
         sess.loginAndCreateSession(targetHost);
         port = sess.port;
         sid = sess.sid;
         String[] list_subscribed = port.getServices(services);
         System.Console.WriteLine("Number of services subscribed = "+
```

```
            list_subscribed.Length);
        foreach ( String service in list_subscribed)
        System.Console.WriteLine("Service subscribed = "+service);
    }
```

Sample Output

The following sample output shows one possible result when running the Connect Session Service example:

```
Test 2
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141018874H3
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
```

Identify Services Example

The Identify Services example code discussed below is located in the `\Test3\IdentifyServicesExample.cs` file.The methods included in this example are described in Table 13.

Table 13: Identify Services Methods

Method Name	Method Description
`execute()`	This method creates a session, checks out a license, and then prints the services license that was retrieved.

This example uses code from the Connect Session Service example to create a session and check out a license for the requested service.

Note: To browse service licenses, you must have already checked out a license for service.

```
using System;

   public class IdentifyServicesExample{
```

execute() Method

This method uses the `ConnectSessionServiceExample` to log in and subscribe to a service. Then it views the available licenses, releases the currently used license, and finally logs out of the session.

```
public void execute(String targetHost)
{
   ConnectSessionServiceExample cs = new ConnectSessionServiceExample();
   cs.execute(targetHost);
```

The `ConnectSessionServiceExample` object log into the GIS server and subscribes to a service, as described on page 84.

```
   String[] licenses_checked = cs.port.browseServices();
   foreach (String service in licenses_checked)
      System.Console.WriteLine("browsing services available:  "+ service);
```

Using the `ConnectSessionServiceExample` object's exposed port, a list of available services is identified and then printed.

```
   System.Console.WriteLine("releasing services...");
   String[] licenses_released = cs.port.releaseServices(cs.services);
   foreach (String service in licenses_released)
      System.Console.WriteLine("Services released:  "+ service);
   cs.port.logout(cs.sid);//logout request
   System.Console.WriteLine("logout from session");
}
```

Using the `ConnectSessionServiceExample` object's exposed port, all services are released. A list of released services is printed, and a logout request is sent to close the session.

Sample Output

The following sample output shows one possible result when running the Identify Services example:

```
Test 3
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141053248H4
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
browsing services available:  GIS_CONFIGSERVICE
releasing services...
Services released:  GIS_CONFIGSERVICE
logout from session
```

Chapter

10 Register to Configuration Service Example—C#

This chapter includes the following sections:

Example Overview

The Register To Configuration Service example demonstrates how to connect, log in, and register to the Session and Configuration interface.

Register To Configuration Service Example

The Register To Configuration Service example code is located in the `\Test4\RegisterToConfigurationExample.cs` file. The methods included in this example are described in Table 14.

Table 14: Register To Configuration Service Example Methods

Method Name	Method Description
`Execute()`	This method logs into GIS and retrieves a session ID. Then it logs in and registers with a Configuration service, before closing the session.

Register To Configuration Service Example Code

The Register To Configuration Service example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.IO;
using System.Xml;
using System.Threading;
using System.Web.Services.Protocols;
using System.Globalization;

   public class RegisterToConfigurationServiceExample
   {
      public string cfgUserName = "default";
      public string cfgUserPassw = "password";
      public string cfgAppName = "default";
      public GCE cfgService;

   public void execute(String targetHost)
   {
      ConnectSessionServiceExample cs = new ConnectSessionServiceExample();
      cs.execute(targetHost);
```

Create a new proxy instance.

```
      cfgService = new GCE();
      cfgService.Url = "http://" + targetHost +
         "/gis/services/CSProxyService?GISsessionId="+cs.sid;
```

Create a cookie container for passing configuration session Id.

```
      cfgService.CookieContainer = new System.Net.CookieContainer();
      Console.Write("==>Connecting and registering at Configuration Service ...");
      cfgService.register(cfgUserName,cfgUserPassw, cfgAppName);
      Console.WriteLine("done");
   }
}
```

Sample Output

The following sample output shows one possible result when running the Register To Configuration Service example:

```
Test 4
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141228386H5
Number of services subscribed = 1
```

```
Service subscribed = GIS_CONFIGSERVICE
==>Connecting and registering at Configuration Service ...done
```

Chapter

11 Get Configuration Information Example—C#

This chapter includes the following section:

Example Overview

This example demonstrates how you use query expressions to retrieve Configuration information from an established Configuration session using `get` requests, and how to process the results of your queries.

Some common query expressions are included as examples in the code. These queries are listed and described in "Query Analysis" on page 95.

Note: All Genesys SDKs use a proprietary query language that is based on a subset of the XPath expression language. Standard XPath queries will usually work, however, not all XPath functionality is supported.

For more information on how to construct queries, see the *Configuration SDK 7.6 Web Services API Reference*.

Get Configuration Information Example

The Get Configuration Information example code discussed is located in the `\Test5\GetConfigurationInformationExample.cs` file. The methods included in this example are described in Table 15 on page 92.

Table 15: Get Configuration Information Example Methods

Method Name	Method Description
`execute()`	This method creates a session, and logs in and registers with the GIS and Configuration services. The `get()` method is used to send queries and return configuration data in XML format.
DisplayResults()	This method displays the query and results that are used as parameters. A query returns `MessageElement` objects reflecting the `update` schema structure.

This sample explains how to accomplish Configuration service `get` requests. It gives examples of different query expressions that can be used, and how to process those queries.

Get Configuration Information Example Code

The Get Configuration Information example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.Xml;

public class GetConfigurationInformationExample
{
```

These are the connection parameters used to connect to and register with GIS and the Configuration service:

```
  // get all Agents
  string queryAllAgents = "CfgAgent";
  // get Agent Groups where specific agent participates
  string queryAgentGroups = "CfgAgent[@name='Lucent_1000']/agentGroups/*/baseDBID";
  // get all Agents from some Agent Group
  string queryAgents = "CfgAgentGroup[@name='Agent Group']/agentDBIDs/*/linkDBID";
  // get all Skills for some Agent
  string querySkills = "CfgAgent[@name='Lucent_1001']/skillLevels/*/linkDBID";
  // get all Switches connected to some Agent (through its Login)
  string querySwitches = "CfgAgent[@name='Lucent_1001']/agentLogins/*
     /linkDBID ownerDBID";
  // get all Agents assigned to a particular Place
  string queryAgentsByPlace = "CfgAgent[placeDBID/@name='Place_1000']";
  // get the all DNs of type 'ACD Position' under some Switch
```

```
   string queryDNs = "CfgSwitch[@name='LucentG3']/DNs/CfgDN[@type='2']";
   // get all the Agents under some Folder
   string queryAgentsByFolder= "CfgFolder[@name='SubFolder']/objectIDs
      /CfgPersonShortcut/linkDBID";
```

execute() Method

```
public void execute(String targetHost)
{
   RegisterToConfigurationServiceExample registerExample =
      new RegisterToConfigurationServiceExample();
   registerExample.execute(targetHost);

   GCE cfgService = registerExample.cfgService;
```

Once the session is established and the Configuration service is registered, the version number is displayed..

```
   Console.WriteLine("==>Configuration Service version is ..." +
      cfgService.getVersion());
```

Each query is sent to the Configuration service using the `get` request, and both the query and results are then displayed using the `DisplayResults()` method.

```
   DisplayResults(queryAllAgents, ((System.Xml.
      XmlNode[]) cfgService.get(queryAllAgents))[0]);

   // read information about all the Agent's Agent Groups.
   DisplayResults(queryAgentGroups,((System.Xml.XmlNode[])cfgService.
      get(queryAgentGroups))[0]);

   // read information about all the Agents from some Agent Group
   DisplayResults(queryAgents,((System.Xml.XmlNode[])cfgService.get(queryAgents))[0]);

   // read information about all the Skills for some Agent
   DisplayResults(querySkills,((System.Xml.XmlNode[])cfgService.get(querySkills))[0]);

   // read information about all the Switches for some Agent
   DisplayResults(querySwitches,((System.Xml.XmlNode[])cfgService.
      get(querySwitches))[0]);

   // read information about all the Agents assigned to a particular Place
   DisplayResults(queryAgentsByPlace,((System.Xml.XmlNode[])cfgService.
      get(queryAgentsByPlace))[0]);

   // read information about all the DNs of type 'ACD Position' under some Switch
   DisplayResults(queryDNs,((System.Xml.XmlNode[])cfgService.get(queryDNs))[0]);

   // read information about all the Agents under some Folder
```

```
DisplayResults(queryAgentsByFolder,((System.Xml.XmlNode[])cfgService.
   get(queryAgentsByFolder))[0]);
```

To close the session:

```
// close session
cfgService.unregister();
}
```

DisplayResults() method

This method displays the query and results that are used as parameters. A query returns `MessageElement` objects reflecting the `update` schema structure.

```
void DisplayResults(string query,XmlElement result)
{
   Console.WriteLine("Results of the query : '{0}'",query);
   foreach (XmlNode node in result.ChildNodes)
      Console.WriteLine(" =>Object [{0}], name [{1}]", node.Name,
         ((XmlElement)node).GetAttribute("name"));
}
```

Sample Output

The following sample output shows one possible result when running the Get Configuration Information example:

```
Test 5
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141374948H12
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
==>Connecting and registering at Configuration Service ...done
==>Configuration Service version is ...7.5.000.11
Results of the query : 'CfgAgent'
 =>Object [CfgAgent], name [Meridian_8000]
 ...
Results of the query : 'CfgAgent[@name='ALU_1000']/agentGroups/*/baseDBID'
 =>Object [CfgAgentGroup], name [Agent Group G3]
 =>Object [CfgAgentGroup], name [Agent Group 80001]
 =>Object [CfgAgentGroup], name [Agent Group Everybody]
Results of the query : 'CfgAgentGroup[@name='Agent Group']/agentDBIDs/*/linkDBID'
Results of the query : 'CfgAgent[@name='ALU_1001']/skillLevels/*/linkDBID'
Results of the query : 'CfgAgent[@name='ALU_1001']/agentLogins/*/linkDBID/ownerDBID'
 =>Object [CfgSwitch], name [AlcatelLucentG3]
Results of the query : 'CfgAgent[placeDBID/@name='Place_1000']'
 =>Object [CfgAgent], name [ALU_1000]
```

```
Results of the query : 'CfgSwitch[@name='LucentG3']/DNs/CfgDN[@type='2']'
Results of the query : 'CfgFolder[@name='SubFolder']/objectIDs/CfgPersonShortcut/
   linkDBID'
```

Query Analysis

The queries used to retrieve Configuration information are described in Table 16. Queries are based on the CSProxy schema definition and use a Genesys proprietary query language that is based on a subset of the XPath query format.

Table 16: List of Queries

Query Description	Query Name	Query
Retrieve all available Agents.	queryAllAgents	CfgAgent
Retrieve all Agent Groups to which a specific agent belongs.	• queryAgentGroups • queryAgentGroupsAlt	• CfgAgentGroup[agentDBIDs/*/linkDBID/@name='<AgentID>'] • CfgAgent[@name='<AgentID>']/agentGroups/*/baseDBID
Retrieve all Agents from a specific Agent Group.	• queryAgents • queryAgentsAlt	• CfgAgentGroup[@name='<AgentGroup>']/agentDBIDs/*/linkDBID • CfgAgent[agentGroups/*/baseDBID/@name='<AgentGroup>']
Retrieve all Skills for a specific Agent.	querySkills	CfgAgent[@name='<AgentID>']/skillLevels/*/linkDBID
Retrieve all Switches connected to a specific Agent through the agent Login.	querySwitches	CfgAgent[@name='<AgentID>']/agentLogins/*/linkDBID/ownerDBID
Retrieve all Agents assigned to a particular Place.	queryAgentsByPlace	CfgAgent[placeDBID/@name='<PlaceName>']
Retrieve all Places with no assigned Agents.	queryPlacesWoAgents	CfgPlace[not agents]

Table 16: List of Queries (Continued)

Query Description	Query Name	Query
Retrieve all DNs of the type ACD Position that belong to a specific Switch.	queryDNs	CfgSwitch[@name='<SwitchName>']/DNs/CfgDN[@type='<DnType>']
Retrieve all Agents in a specific Folder.	queryAgentsByFolder	CfgFolder[@name='<FolderName>']/objectIDs/CfgPersonShortcut/linkDBID

Chapter

12 Update Configuration Data Example—C#

This chapter includes the following section:

Example Overview

This example uses `update` requests to modify Configuration service information using by uploading an XML data file, and shows how to process the results of the request. The actual sequence of updates is contained in a separate XML data file (`/SRC/sample_01.xml`) for readability.

Update Configuration Data Example

The Update Configuration Data example code is located in the `\Test6\UpdateConfigurationDataExample.cs` file. The methods included in this example are described in Table 17 on page 97.

Table 17: Update Configuration Data Methods

Method Name	Method Description
`execute()`	This method creates a session, and then logs in and registers with GIS and Configuration services. It then reads the XML update sequence, sends an `update` request to the Configuration Server, and displays the response.

Update Configuration Data Example Code

The Update Configuration Data example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.IO;
using System.Xml;

   public class UpdateConfigurationDataExample
   {
      // set of XML data files composing a sequence of updates
      string [] dataFiles    = {"sample_01.xml"};
```

Specify the parameters used to connect to GIS and register with the Configuration service, along with the location of the XML data file holding the update sequence.

execute() Method

This method creates a session, and then logs in and registers with GIS and Configuration services.

```
   public void execute(String targetHost)
   {
      try
      {
         RegisterToConfigurationServiceExample registerExample =
            new RegisterToConfigurationServiceExample();
         registerExample.execute(targetHost);
         GCE cfgService = registerExample.cfgService;

         // get Configuration Service version
         Console.WriteLine("==>Configuration Service version is ..." +
            cfgService.getVersion());
```

Once the session is established and the Configuration service is registered, the version number is displayed.

```
         foreach (string file in dataFiles)
         {
            XmlDocument doc = new XmlDocument();
            doc.Load(file);
```

Load the XML data file.

```
            Console.Write("==>Updating configuration using {0}...",file);
            XmlElement result = (XmlElement)cfgService.
```

```
            update((System.Xml.XmlElement)doc.DocumentElement);
         Console.WriteLine("done");
```

Use the Configuration service `update` request, and keep the results.

```
         Console.WriteLine("===>Results :");
         foreach (XmlElement item in result.ChildNodes)
            if (item["errorInfo"].GetAttribute("returnCode") == "0")
               Console.WriteLine(" Item : id [{0}] processed successfully",
                  item.GetAttribute("id"));
            else
               Console.WriteLine(" Item : id [{0}] processed with error : {1}",
                  item.GetAttribute("id"),
                  item["errorInfo"]["serverErrorInfo"]["description"].InnerText);
      }
      Console.WriteLine("done");
      cfgService.unregister();
   }
```

Print the results from the update and close the session.

```
   catch (Exception e)
   {
      Console.WriteLine(e.Message);
   }
}
```

Sample Output

The following sample output shows one possible result when running the Update Configuration Data example:

```
Test 6
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141558267H19
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
==>Connecting and registering at Configuration Service ...done
==>Configuration Service version is ...7.5.000.11
==>Updating configuration using sample_01.xml...done
===>Results :
 Item : id [Skill-001] processed successfully
 Item : id [Skill-002] processed successfully
 Item : id [Environment] processed successfully
done
```

Chapter

13 Solicited Notification Examples—C#

This chapter includes the following sections:

Solicited Notification Examples Overview

Two solicited notification examples are provided to demonstrate the difference between Notification by Polling and Notification by Blocking:

- Solicited Notification by Polling—This example shows how to use polling to access the Configuration service notification mechanism. When polling, you subscribe to a Configuration service and then use the `refresh()` request to check for updates.
- Solicited Notification by Blocking—This example shows how to use asynchronous calls to the `getEvent` method to access the Configuration service notification mechanism.

This example uses XML data stored in a separate file (`\src\sample_01x.xml`) for readability. The XML file contains information to create a new Skill object, allowing you to test the notification process.

Solicited Notification by Polling Example

The Solicited Notification by Polling example code is located in the `\Test8\SolicitedNotificationByPollingExample.cs` file. The methods included in this example are described in Table 18.

Table 18: Solicited Notification by Polling Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then subscribes to receive notification about Agent Skills, and polls for updates.
`DisplayNotification()`	This method displays the response returned after using `refresh` to poll for notification updates.

Solicited Notification by Polling Example Code

The Solicited Notification by Polling example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.IO;
using System.Xml;
   public class NotificationByPollingExample
   {
```

DisplayNotification() Method

Use this method to display the notification results after using the `refresh()` request.

```
      void DisplayNotification(XmlElement cfgData,string elementName,string textStr)
      {
         XmlElement operElem = cfgData[elementName];
         if (operElem != null && operElem.ChildNodes.Count > 0)
         {
            Console.WriteLine(" ===>" + textStr + ":");
            foreach (XmlElement elem in operElem.ChildNodes)
               Console.WriteLine("  Object [{0}], DBID [{1}], name [{2}]",
                  elem.Name,elem.GetAttribute("DBID"),elem.GetAttribute("name"));
         }...
```

execute() Method

This method creates a session, logs into a GIS server, and subscribes to the licensed services.

```
public void execute(String targetHost)
{
   try
   {
      RegisterToConfigurationServiceExample registerExample =
         new RegisterToConfigurationServiceExample();
      registerExample.execute(targetHost);

      GCE cfgService = registerExample.cfgService;

      // get Configuration Service version
      Console.WriteLine("==>Configuration Service version is ..." +
         cfgService.getVersion());
```

Initialize the Configuration service helper class, log onto GIS and register with the Configuration service, and display the Configuration service version.

```
      Console.WriteLine("==>Subscription on notifications completed ..." +
         cfgService.subscribe("CfgSkill","",""));
```

Subscribe to receive update notifications for Skill configuration objects.

```
      object []obj = (object [])cfgService.refresh("");
      string checkpoint = ((XmlElement)obj[0]).InnerText;
      Console.WriteLine("===>Current checkpoint is [{0}]",checkpoint);
```

Use the `refresh()` request to get the current checkpoint.

```
      XmlDocument doc = new XmlDocument();
      doc.Load("sample_01x.xml");
      Console.Write("==>Updating configuration using sample_01x.xml...");
      XmlElement result = (XmlElement)cfgService.update(doc.DocumentElement);
      Console.WriteLine("done");
```

Read the XML document from the `sample_01x.xml` file, and use the `update()` request to modify the configuration data.

```
      obj = (object [])cfgService.refresh(checkpoint);
      checkpoint = ((XmlElement)obj[0]).InnerText;
      Console.WriteLine("===>Current checkpoint is [{0}]",checkpoint);
      XmlElement elem = (XmlElement)obj[1];
      DisplayNotification(elem,"CfgCreate","Created");
      DisplayNotification(elem,"CfgUpdate","Updated");
```

```
            DisplayNotification(elem,"CfgRemove","Removed");
```

Refresh your checkpoint and use the DisplayNotification() method to display the updated Skill that was returned.

```
            cfgService.unregister();
         }
         catch (Exception e)
         {
            Console.WriteLine(e.Message);
         }...
```

Solicited Notification by Blocking Example

The Solicited Notification by Blocking example code is located in the `\Test7\SolicitedNotificationByBlockingExample.cs` file. The methods included in this example are described in Table 19.

Table 19: Solicited Notification by Blocking Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then subscribes to receive notification about Skills and waits for an update.
`DisplayNotification()`	This method displays the response returned to the `GetEventCallback()` method.
`GetEventCallback()`	This method is the required asynchronous `getEvent()` method callback function.

Solicited Notification by Blocking Example Code

The Solicited Notification by Blocking example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.IO;
using System.Xml;
using System.Runtime.InteropServices;
using System.Web.Services.Protocols;

   public class SolicitedNotificationByBlockingExample
   {
```

GetEventCallback() Method

This is the asynchronous `getEvent` method `callback` function.

```
void GetEventCallback(IAsyncResult ar)
{
   // perform an EnfgetEvent call
   object []objs = (object[])((GCE)ar.AsyncState).EndgetEvent(ar);
   XmlElement elem = (XmlElement)objs[0];
   Console.WriteLine("==>GetEvent response received.Results :");
   DisplayNotification(elem,"CfgCreate","Created");
   DisplayNotification(elem,"CfgUpdate","Updated");
   DisplayNotification(elem,"CfgRemove","Removed");
}
```

DisplayNotification() Method

Use this method to display notification results.

```
void DisplayNotification(XmlElement cfgData,string elementName,string textStr)
{
   XmlElement operElem = cfgData[elementName];
   // display only if there's something to display
   if (operElem != null && operElem.ChildNodes.Count > 0)
   {
      Console.WriteLine(" ===>" + textStr + ":");
      foreach (XmlElement elem in operElem.ChildNodes)
         Console.WriteLine("  Object [{0}], DBID [{1}], name [{2}]",
            elem.Name,elem.GetAttribute("DBID"),elem.GetAttribute("name"));
   }
}
```

execute() Method

The execute() method accomplishes all the example application logic.

```
public void execute()
{
   try
   {
      RegisterToConfigurationServiceExample registerExample =
         new RegisterToConfigurationServiceExample();
      registerExample.execute(targetHost);

   GCE cfgService = registerExample.cfgService;

   // get Configuration Service version
```

```
        Console.WriteLine("==>Configuration Service version is ..." +
          cfgService.getVersion());
```

Initialize the Configuration service helper class, log onto GIS and register with the Configuration service, and display the Configuration service version.

```
          Console.WriteLine("==>Subscription on notifications completed ..." +
            cfgService.subscribe("CfgAgent","",""));
          Console.WriteLine("==>Initiate notification request");
          IAsyncResult asyncResult = cfgService.BegingetEvent(new
            AsyncCallback(GetEventCallback),cfgService);
```

Subscribe to notifications about Agents, and then initialize an asynchronous `getEvent` call.

```
          while (asyncResult.AsyncWaitHandle.WaitOne(50,true) == false)
          {
            // do something ...
          }
```

Use a loop to wait for the response. The `GetEventCallback` method for this example displays the results when they are returned; no additional code is required here.

```
          Console.WriteLine("done");
          cfgService.unregister();
        }
        catch (Exception e)
        {
          Console.WriteLine(e.Message);
        }
    }
```

Sample Output

The following sample output shows one possible result when running the Solicited Notification by Blocking example:

```
Test 7
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141658262H27
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
==>Connecting and registering at Configuration Service ...done
==>Configuration Service version is ...7.5.000.11
==>Subscription on notifications completed ...OK
==>Initiate notification request
```

```
done
==>GetEvent response received.Results :
 ===>Updated:
  Object [CfgAgent], DBID [237], name [ALU_2000]
```

Chapter

14 Unsolicited Notification Example—C#

This chapter includes the following sections:

Unsolicited Notification Overview

This example explains how to work with the Configuration service notification mechanism by using unsolicited notifications.

The Unsolicited Notification method in the configuration example sends a request to GIS for automatic updates on a configuration object to which you have already subscribed. This example shows how to use the Remoting feature available in Microsoft .NET to receive update messages from GIS.

Unsolicited Notification Example

The Unsolicited Notification example code is located in the `\Test9\UnsolicitedNotificationExample.cs` file. The methods included in this example are described in Table 20 on page 110.

Table 20: Unsolicited Notification Methods

Method Name	Method Description
`DisplayNotification()`	This method displays responses returned using the `notifyMessage()` method.
`notifyInitMessage()`	This method is called immediately after the subscribeEx() request is made.
`notifyMessage()`	Called whenever a new notification message arrives.
`execute()`	This method creates a session, and then logs in and registers with GIS and Configuration services. It then subscribes to services, registers a channel to listen for responses on, and waits for notifications to arrive.

Unsolicited Notification Example Code

The Unsolicited Notification example code is listed below, with comments and analysis added to focus your attention on important details or clarify complex procedures.

```
using System;
using System.IO;
using System.Xml;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
```

NotifyService Class

The Notify Service class methods are called whenever a notification arrives at the listening port.

```
public class NotifyService : MarshalByRefObject
{
```

DisplayNotification() Method

Display the results of the notification.

```
    private void DisplayNotification(XmlElement cfgData,string elementName,
       string textStr)
```

```
      {
         XmlElement operElem = cfgData[elementName];
         // display only if there's something to display
         if (operElem != null && operElem.ChildNodes.Count > 0)
         {
            Console.WriteLine(" ===>" + textStr + ":");
            foreach (XmlElement elem in operElem.ChildNodes)
               Console.WriteLine("  Object [{0}], DBID [{1}], name [{2}]",
                  elem.Name,elem.GetAttribute("DBID"),elem.GetAttribute("name"));
         }
      }
```

notifyInitMessage() Method

Call the `notifyInitMessage()` method immediately after making the `subscribeEx` call.

```
   public void notifyInitMessage(string serverHost, string serverPort,
      string serverApp)
   {
      Console.WriteLine("Init message received from {1}:{0},
         application [{2}]",serverHost,serverPort,serverApp);
   }
```

notifyMessage() Method

The `notifyMessage()` method is called whenever a new notification message arrives.

```
   public void notifyMessage(string result)
   {
      XmlDocument doc = new XmlDocument();
      doc.LoadXml(result);
      DisplayNotification(doc.DocumentElement,"CfgCreate","Created");
      DisplayNotification(doc.DocumentElement,"CfgUpdate","Updated");
      DisplayNotification(doc.DocumentElement,"CfgRemove","Removed");
   }
}
```

UnsolicitedNotification Class

The `UnsolicitedNotificationExample` class is the main working class for the application.

```
   public class UnsolicitedNotificationExample
   {
   int notifyPort = 8001;
   string notifyHost = System.Net.Dns.GetHostName();
```

```
string microsoftNS = "http://schemas.microsoft.com/clr/nsassem";
string serviceClass = "NotifyService";
```

These are the connection parameters used to connect and register to GIS and Configuration service.

execute() Method

The `execute()` method accomplishes all the sample's application logic.

```
public void execute(String targetHost)
{
   try
   {
      RegisterToConfigurationServiceExample registerExample =
         new RegisterToConfigurationServiceExample();
      registerExample.execute(targetHost);

      GCE cfgService = registerExample.cfgService;

      // get Configuration Service version
      Console.WriteLine("==>Configuration Service version is ..." +
         cfgService.getVersion());
```

Initialize the Configuration service, log onto GIS and register with the Configuration service, and then display the Configuration service version.

```
      // register channel
      HttpServerChannel channel = new HttpServerChannel("notifyChannel",notifyPort);
      ChannelServices.RegisterChannel(channel);
      RemotingConfiguration.RegisterWellKnownServiceType
         (typeof(NotifyService), "notifyService", WellKnownObjectMode.Singleton);
```

Register the new HTTP server channel that is used for notification.

```
      XmlDocument doc = new XmlDocument();
      XmlElement subscriptionData = doc.CreateElement("","subscriptionData","");
      XmlElement subscription = doc.CreateElement("","subscription","");
      subscription.SetAttribute("type","","CfgAgent");
      subscriptionData.AppendChild(subscription);
      doc.AppendChild(subscriptionData);
```

Next, subscribe for update notifications on whatever objects interest you. This example shows how to receive updates on skills.

The `NotifyService` class, described earlier, is now responsible for processing all notifications received.

```
        // subscribe on notifications on Skills
        Console.WriteLine("==>Subscription on notifications completed ..." +
           cfgService.subscribeEx(doc.DocumentElement,
           "http://" + notifyHost + ":"+notifyPort+"/notifyService",
           microsoftNS+"/"+serviceClass+"/Test9#"));

        Console.WriteLine("\n Listening for notifications ... (press Enter to finish)
   \n");
        Console.ReadLine();

        // now NotifyService class is responsible for processing of the notifications

        Console.WriteLine("done");
        cfgService.unregister();

     }
     catch (Exception e)
     {
        Console.WriteLine(e.Message);
     }
   }
}
```

Sample Output

The following sample output shows one possible result when running the Unsolicited Notification example:

```
Test 9
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221141809680H29
Number of services subscribed = 1
Service subscribed = GIS_CONFIGSERVICE
==>Connecting and registering at Configuration Service ...done
==>Configuration Service version is ...7.5.000.11
==>Subscription on notifications completed ...

 Listening for notifications ... (press Enter to finish)

Init message received from FRBRESV158668:2025, application [confserv]
 ===>Updated:
  Object [CfgAgent], DBID [237], name [ALU_2000]
```

Index

H

J

L

M

N

O

P

Q

R

S

T

V

W

X