Statistics 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_stat-ws_09-2008_v7.6.101.00

Table of Contents

Preface

Welcome to the *Statistics SDK 7.6 Web Services Developer's Guide.* This guide presents an overview of GIS architecture and communication protocols, setup procedures for client development, and examples of code for accessing and using statistics from Stat Server via the Genesys Integration Server (GIS).

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

The Statistics SDK Service provides you with access to the Statistics API of the Genesys Integration Server (GIS), which enables you to access Stat Server statistical information from your own web-based client application. The Statistics 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 Statistics SDK Service.

Intended Audience

This guide, primarily intended for developers who are familiar with Hypertext Transfer Protocol (HTTP) and XML (Extensible Markup Language) technologies, 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
- SOAP Protocol
- WSDL (Web Services Description Language)

Depending on the technology choice for client development, working knowledge of Java or of a Web Services client-side programming language may be required. Developers should be familiar with the Genesys Framework, especially with the Configuration Layer and Stat Server.

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 these 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, demonstrates how to request and receive a single set of data on a specified statistic.
- Chapter 5, explains how to request information about a statistic, including the statistic names available for subscription, time range, and time profile, if available.
- Chapter 6, explains how to obtain repeated updates on statistics to which you have subscribed. You can solicit the updates using the Polling or Blocked notification modes.
- Chapter 7, provides an example of how to construct a client server that can receive automatic updates from GIS concerning statistics to which you have subscribed.
- Chapter 8, includes three examples of how to use the Session Service to log in, verify licensing, and log out.
- Chapter 9, explains how to request information about a statistic, including the statistic names available for subscription, time range, and time profile, if available.
- Chapter 10, demonstrates how to request and receive a single set of data on a specified statistic.
- Chapter 11, explains how to obtain repeated updates on statistics to which you have subscribed. You can solicit the updates using the Polling or Blocked notification modes.
- Chapter 12, provides an example of how to construct a client server that can receive automatic updates from GIS concerning statistics 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 *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.

- The *Genesys Integration Server 7.6 Deployment Guide,* which provides installation, configuration, and starting and stopping instructions for GIS.
- The *Configuration SDK 7.6 Web Services Developer's Guide,* which explains how to set up the Configuration SDK code examples, including those that provide read-only configuration information for users with a Statistics SDK Web Services license only, and presents the text of the examples along with explanatory comments.
- 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 *Framework 7.x Stat Server User's Guide,* which contains instructions on using Stat Server. See especially the "Custom Value Statistic Types" section.
- 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 Statistics SDK Service

The Genesys Statistics SDK Service product provides developers with the Genesys Integration Server (GIS), which presents the Statistics SDK Service, as well as developer essentials, such as documentation and code examples, to assist you in creating a statistics-gathering application.

This chapter presents the following topics:

Statistics SDK Service Overview

This product consists of these elements:

Genesys Integration Server (GIS)—Provides the Session Service and the Statistics SDK Service.

- 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, sent via HTTP, wrapped using the programming language of your choice. The examples included in this SDK 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 document and also in the *Statistics SDK 7.6 Web Services API Reference.*

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

The Statistics SDK Service—Provides access to statistical information gathered by Stat Server. Your client application can subscribe (register) itself as interested in a particular sequence of statistical events over time. For example, you might repeatedly get updated information about interactions waiting in queues, agent group activities, or actions of particular agents. You can qualify this sequence by time or change criteria. For such continuous measurements, Stat Server sends update notifications to GIS, which acts, in this case, as a client of Stat Server.

The Configuration SDK Service—Provides access to configuration information through a subset of the Configuration SDK Service methods: `register`, `get`, `getex`, and `getVersion`. This effectively provides read-only access to configuration data for users with only a Statistics SDK Service license. Unless you also have a Configuration SDK Service license, you cannot use any other Configuration SDK Service methods.

The Interaction Services—Enable you to develop custom applications for handling inbound and outbound interactions on voice, e-mail, and/or chat media. You need a separate license to use the Interaction Services.

Note: See the *Statistics SDK 7.6 Web Services API Reference,* the *Configuration SDK 7.6 Web Services Developer's Guide,* and the *Agent Interaction SDK 7.6 Java Developer's Guide* for more information on these methods, and for code examples.

Architecture

Figure 1 on page 17 illustrates the relationships among a statistics-gathering client using the optional notification service, GIS, and the Genesys Framework.

Note: For more information on the notification service, which is used for unsolicited notification, see "Unsolicited Notification Mode" on page 24.

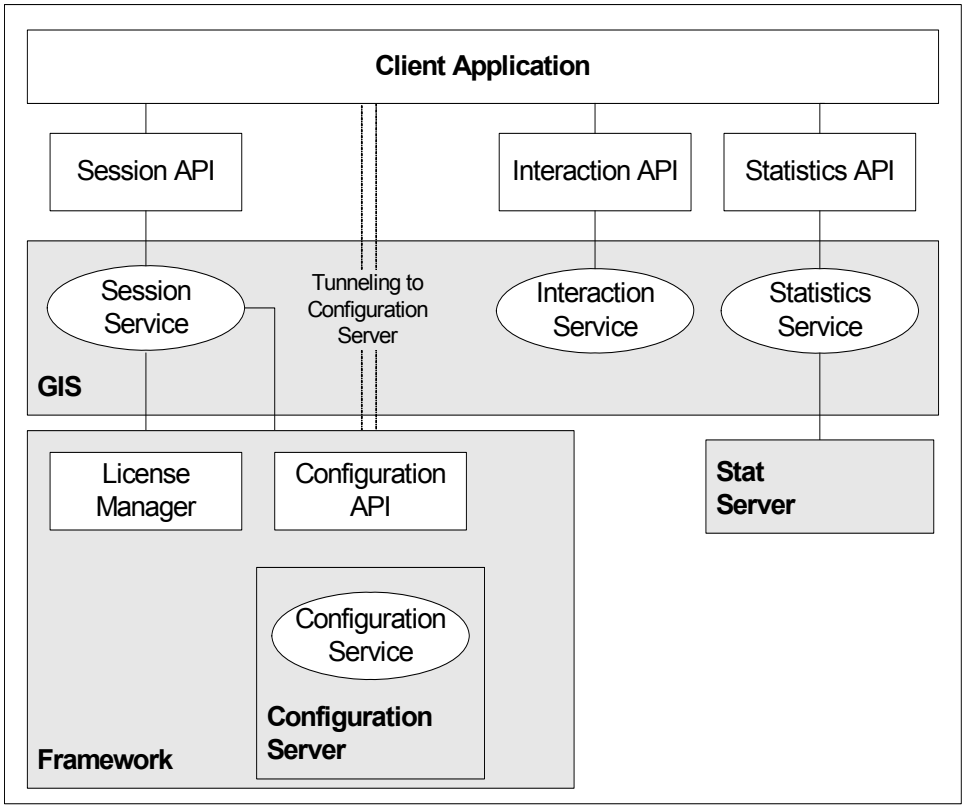

Figure 1: GIS Architecture and Connections to the Genesys Framework

Additional Framework Components

In addition to Stat Server, GIS works with these components of the Genesys Framework:

License Manager—GIS, through the Session Service, uses License Manager to access the license information stored in the license file.

Configuration Server/CS Proxy—GIS accesses Configuration Server's SOAP API for configuration information. This SOAP interface can receive messages sent by your client application and passed on by GIS. You can send these messages to a Configuration Server configured either as a master Configuration Server or as a CS Proxy.

Note: If you are using Configuration Server 6.5, the SOAP interface is presented only in CS Proxy configuration.

For a master Configuration Server, you must specify the SOAP port in Configuration Server's own configuration file, `confserv.cfg`. For a CS Proxy, you must configure the SOAP interface option in the Configuration Server Application object.

Management Layer—You can use Management Layer to view 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). Through LCA integration, you can also use Management Layer to start or stop GIS.

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

Supported Genesys Framework Versions

The Genesys Statistics SDK Service is compatible with Stat Server 6.5 and 7.*x*, Configuration Server Proxy (CS Proxy) 6.5, and Configuration Server 7.*x* (whether configured as a master Configuration Server or as a CS Proxy).

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.

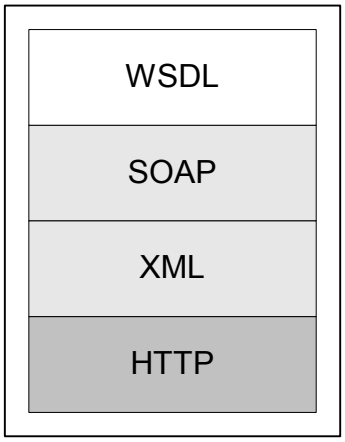

Figure 2: GIS Communication Protocols

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 Statistics SDK Service uses several WSDL files, one each for the Session, Statistics, and Configuration Services; as well as the `notification.wsdl` file, which is used for implementing the Unsolicited notification mode.

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

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 statistical data, the client sends a request for that particular statistic and then receives a response from GIS.

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

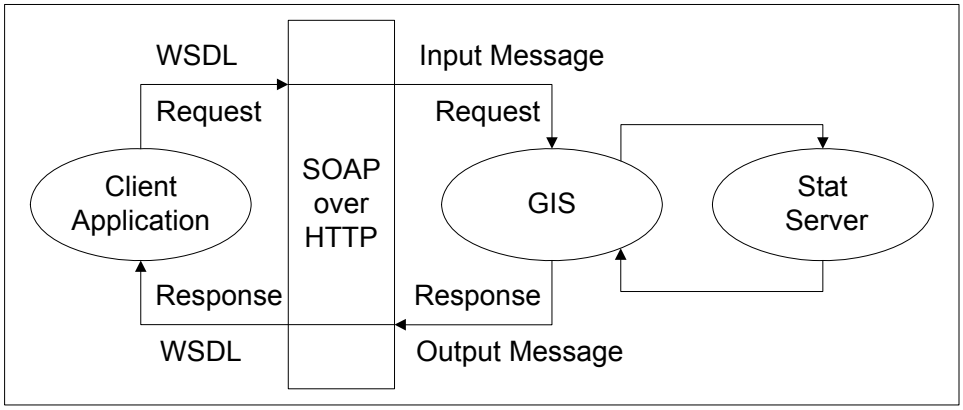

Figure 3: GIS Request/Response Schema

Figure 4 on page 20 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.

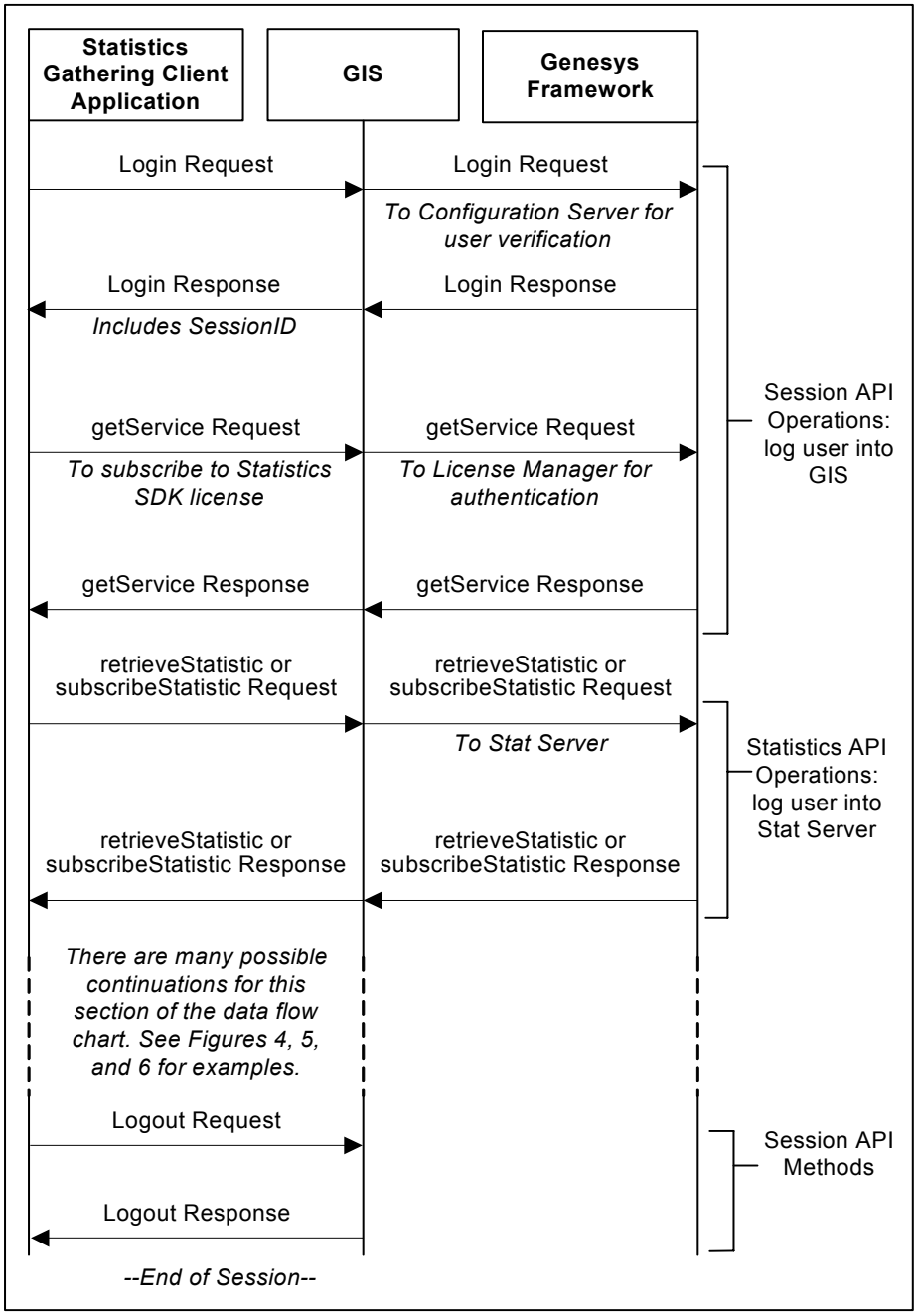

Figure 4: Statistics SDK Service Data Flow

The stages of the data flow are also described in the following sequence.

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 using a user name and password that are registered with Configuration Server via the Session Service. The Session Service also enables licensing authentication.

The login response returned from GIS to your client application includes the Session ID, which must be attached to all subsequent messages for this transaction. This session remains open until it is explicitly closed by the client or until expiration of a configurable timeout without client-server interactions.

2. GIS responds.

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

3. The client application must now subscribe to the `GIS_STATSERVICE` 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* a statistic for an immediate look at the statistic values at the current time or *subscribe* to a statistic for ongoing statistic updates over time.

   You can receive data using one of the two main notification types, solicited or unsolicited.

Note: For best performance, it is recommended that several statistics subscriptions should be retrieved in a single `retrieveSubscribedStatistics` request.

Notification Modes

The Statistics SDK Service supports both solicited and unsolicited notification.

Solicited notification requires you to make a request every time you retrieve a statistic value, and includes both the polling and blocked notification modes. These modes are recommended when your application only needs to check the current value of a statistic, or when you do not require continual updates of the statistic value. Polling notification is the default mode of solicited notification, and is the only option available when you retrieve a statistic value without subscribing to that statistic.

Unsolicited notification uses a different method, and is set in the `subscribeStatistic` operation. Applications that track real-time updates to statistic values should use the unsolicited notification method.

Solicited Notification Modes

In solicited notification modes, your client must request data from GIS each time you want to check a statistic value. The two solicited notification modes are:

- **Polling**—Your client application sends a message requesting a statistic value, and GIS responds immediately by returning the most recent value.

  This method of notification ensures a quick response, but may not accurately track the changes to a statistic value over time.

  When you use the `retrieveStatistic` request to check a single statistic value without creating a subscription, polling is the only notification mode available. Polling is also the default mode for requests that check the value of a subscribed statistic, and can be explicitly specified by setting the notification mode in your `retrieveSubscribedStatistics` request.
- **Blocked**—Your client application sends a message requesting a statistic value, and GIS holds the request until it receives an update from Stat Server. Only then does GIS return the statistic value to your client application.

  Note: Until you receive the updated statistic, GIS consumes one thread. Additionally, until the update arrives, this mode blocks one HTTP 1.1 connection for the client application.

  This method of notification will always report the latest update to a statistic value, but can result in a delayed response if the statistic value does not change frequently. Using blocked notification is ideal when you want to monitor changes to a particular statistic value, but any code that uses this mode should send statistic requests from a separate thread to ensure that processing can continue while waiting for a response.

  Note: By default, there can be only two concurrent-HTTP 1.1 connections between the client and GIS. For details on increasing this value, see the Micrsoft.com .NET `ServicePointManager.DefaultConnectionLimit` property description.

  Blocked notification is only available for statistics that you have subscribed to, and must be specified by setting the notification mode of your `retrieveSubscribedStatistics` request.

The following figures show data flows for solicited notification modes. Note that there are no changes in the data flow between polling and blocked methods—the only difference is the length of time required to receive a response from GIS.

Figure 5 on page 23 shows the data flow that occurs when you have subscribed to a statistic. Figure 6 on page 24 shows the data flow that occurs when you poll for a statistic value using the `retrieveStatistics` request.

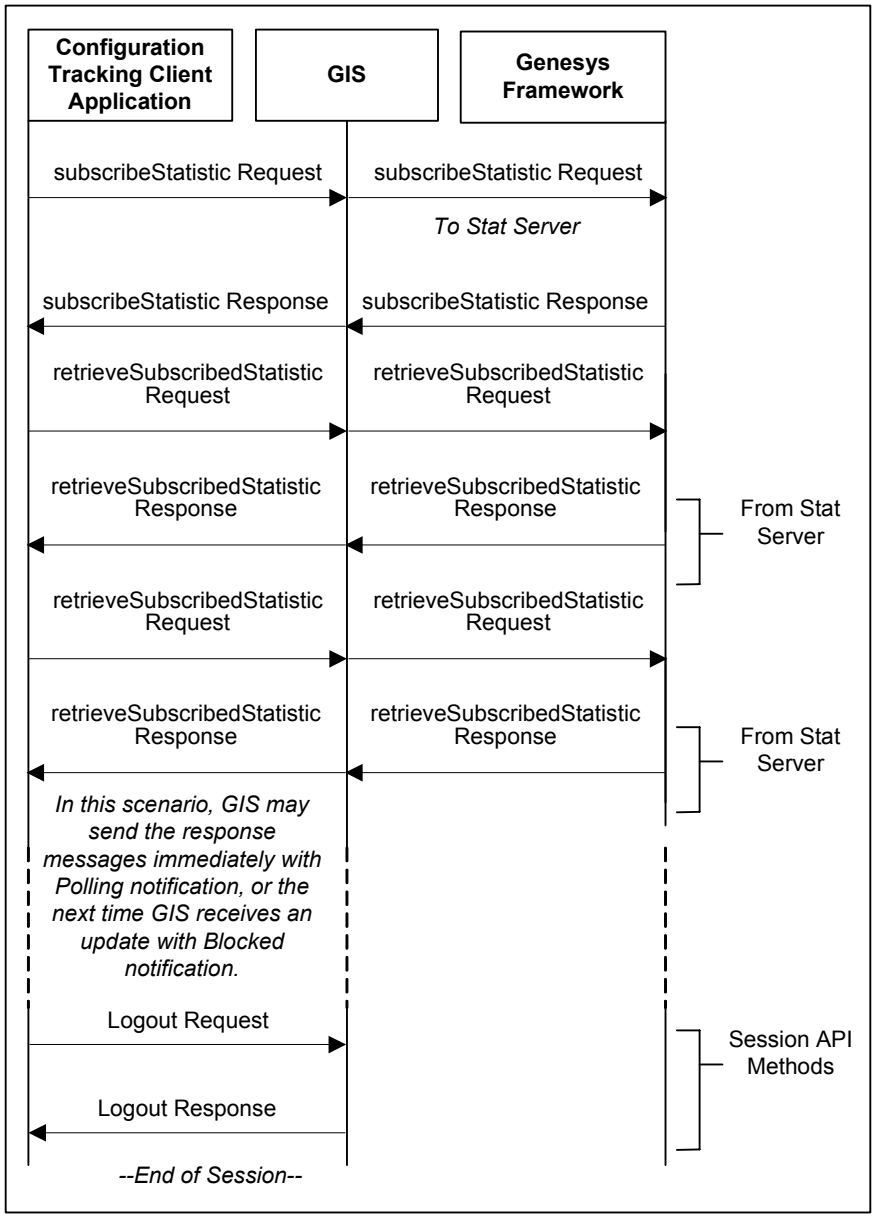

Figure 5: Subscribed Statistics Data Flow

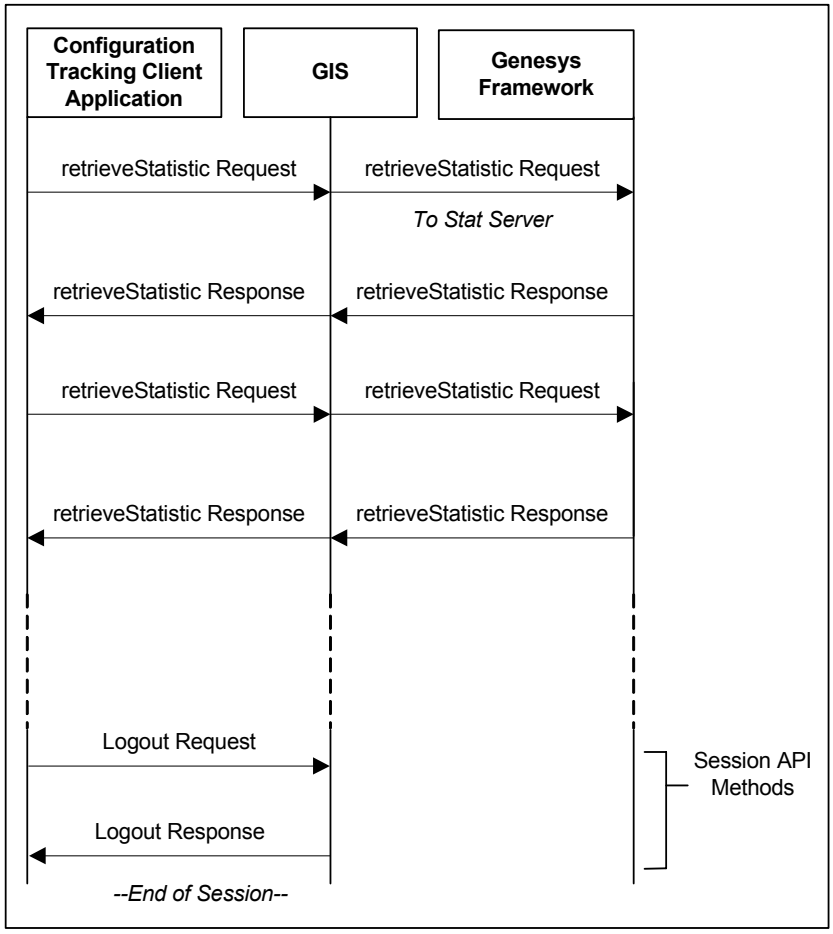

Figure 6: Retrieve Statistics Data Flow

Unsolicited Notification Mode

GIS also provides an unsolicited notification service feature for the Statistics SDK Service. Client applications that use this feature can receive notification of changes to their subscribed statistics without sending requests.

This feature requires a designated client-side HTTP server that allows GIS to transmit statistics updates. You can display these updates on a console window on the server, or create a service that allows client applications to retrieve the statistics updates directly from your client-side server.

Note: The term *client-side server* refers to the HTTP server set up to receive notifications from GIS. The Unsolicited Notification examples included with the Statistics SDK Service documentation require that your client-side server run on Tomcat and AXIS.

Unsolicited notification is only available for statistics you have subscribed to, and it takes precedence over Polling or Blocked modes. The benefit of

unsolicited notification is that your application receives notification of changes to subscribe statistic values as soon as they are updated.

In the data flow for Unsolicited notification mode, shown in Figure 7, GIS sends automatic SOAP messages every time a subscribed statistic has its value updated.

Note: To clarify event sequences, GIS SOAP responses for unsolicited notification in the Statistics SDK Service include timestamps in milliseconds rather than seconds, and they are sent to the client in the order they arrive from Stat Server.

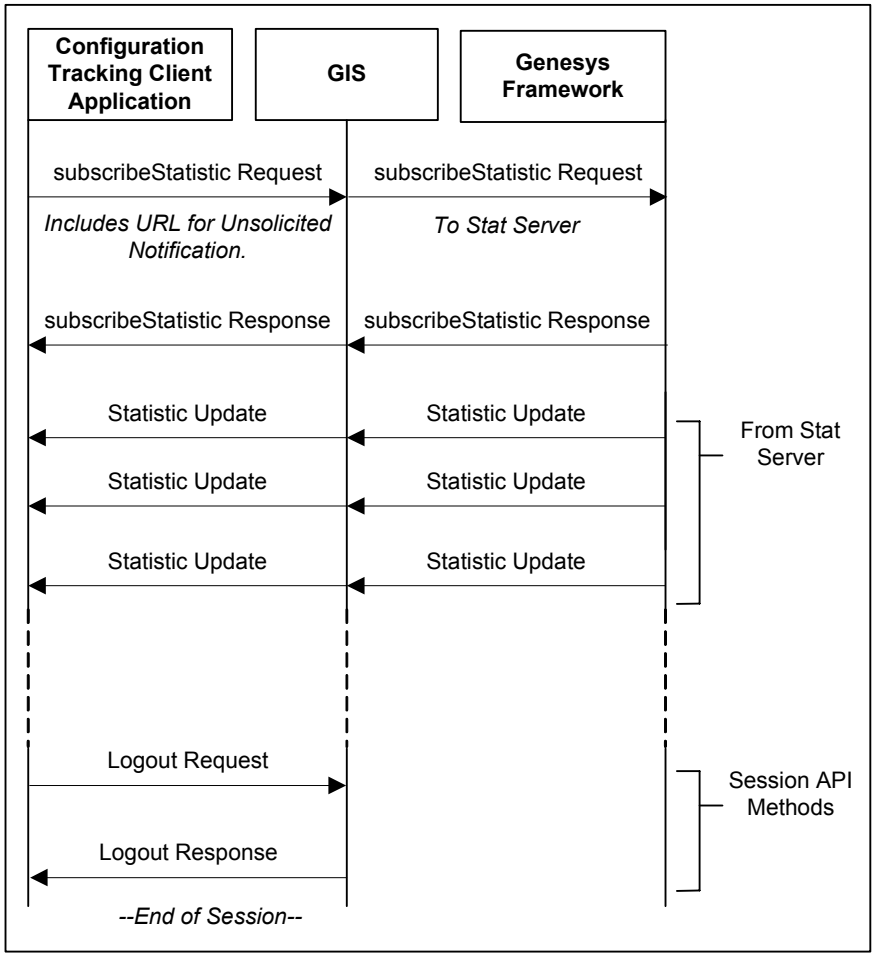

Figure 7: Unsolicited Notification Data Flow

The automatic update messages from GIS are received and interpreted by your client-side HTTP server. The communication parameters for unsolicited notification are described in the Notification WSDL file.

Note: Unsolicited notification mode requires you to subscribe to statistics.

You can customize the notification mechanism by editing the GIS configuration file `modules.conf`. For details, see the "Customize or Uninstall GIS" chapter in your *Genesys Integration Server 7.6 Deployment Guide.*

The examples included in this release provide an example of one way you might implement the Unsolicited Notification feature. They show how a client application can draw updated statistical information from your client-side HTTP server using the following steps:

1. A client application sends a statistic subscription request, together with the URL of a client-side HTTP server, to GIS.
2. Whenever the subscribed statistic is updated, GIS sends statistical update data to the client-side HTTP server, which contains the published interface for the notification service (`NotifyService.java` in this example).
3. The client-side HTTP server saves the received data into a cache and outputs the update to the Tomcat console.
4. Client applications can retrieve that updated statistical information stored in cache by calling the notification service provided by your client-side HTTP server.

Note: The client-side code should filter the notification messages based on the current `sessionId` to filter out previously-cached events.

Note: The client application receiving updates from the client-side HTTP server does not need to be the same application that created the subscription request, as long as the subscription ID is known.

You can see the architecture used for this unsolicited notification example in Figure 8 on page 27.

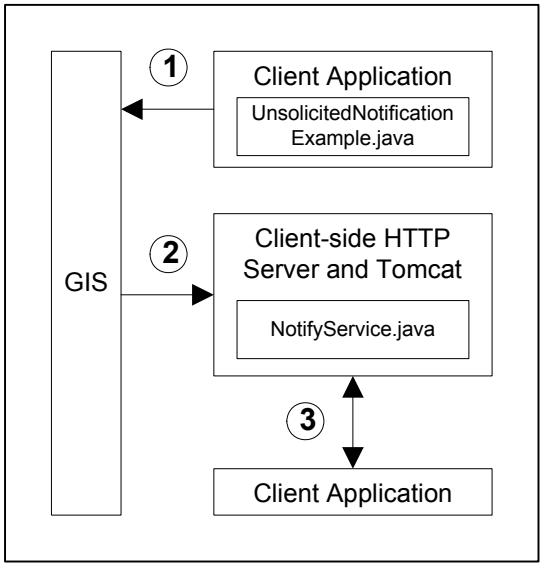

Figure 8: Architecture for Unsolicited Notification

For additional details about unsolicited notification and sample code, see one of the following chapters:

- Unsolicited Notification Examples—Java: Chapter 7 on page 81
- Unsolicited Notification Examples—C#: Chapter 12 on page 119

About the Code Examples

The Statistics 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. The discussion of the Statistics SDK Service examples continues through Chapter 7.
- Discussion of the C# code examples starts with Chapter 8, which presents the code examples specific to the Session Service. The discussion of the Statistics SDK Service examples continues through Chapter 12.

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.

Chapter

2 About the Examples

Documentation for the Statistics SDK Service includes 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 7; C# examples begin in Chapter 8 and continue through Chapter 12.

This chapter contains the following sections:

Examples Overview

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

The code examples show how to create session and statistic requests. Session requests include logging in users, logging out users, and license validation. Statistic requests include actions such retrieving a statistic for an agent, retrieving settings for statistics from Stat Server, and requesting that GIS send statistics automatically to a client web server.

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

Table 1 outlines the relationships among the Java examples. Depending on what your client application needs to do, you must implement at least some (if not all) of the functionality shown in the following dependent levels.

Table 1: Session and Statistics Example Hierarchy

API Feature Examples	Example Filename
Level 1: Create a Session	CreateSessionExample.java
Level 2: Connect Session Service	ConnectSessionServiceExample.java
Level 3: Browse Services	IdentifyServicesExample.java
Level 2 + Retrieve Statistic	RetrieveStatisticExample.java
Level 2 + Retrieve Statistical Profile	RetrieveStatisticalProfileExample.java
Level 4: Subscribe to Statistics	SubscribeToStatisticExample.java
Level 4 + Retrieve Statistic	RetrieveSubscribedStatisticExample.java
Level 2 + Unsolicited Notification	UnsolicitedNotificationExample.java

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 Statistics SDK Service. You do not need a separate license to use the Session Service. After you have checked out a Statistics SDK Service license using the Session Service, your client application can send requests to the Statistics 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 Statistics 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 Statistic examples.
- `ConnectSessionServiceExample.java`—Checks out a service license.
- `IdentifyServiceExample.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 Statistics Service Examples

To retrieve a statistic, your application must make a request for or check out a statistic, and then analyze the response.

The statistic examples demonstrate how to subscribe to a statistic, retrieve a statistic, and request unsolicited notification services from GIS. By reviewing the example code, you will learn how to create the dependent or necessary objects to accomplish these tasks.

Statistics Service Example Files

There are five statistic example files included with this product:

- `RetrieveStatisticExample.java`—Retrieves the current value for a statistic. Combines code from the `SubscribeToStatisticExample.java` and the `RetrieveSubscribedStatisticExample.java` files. See Chapter 4 for details.
- `RetrieveStatisticalProfileExample.java`—Retrieves the parameters for a specified statistic. See Chapter 5 for details.

- `SubscribeToStatisticExample.java`—Subscribes to statistics. See Chapter 6 for details.
- `RetrieveSubscribedStatisticExample.java`—Retrieves values for statistics that you have subscribed to. See Chapter 6 for details.
- `UnsolicitedNotificationExample.java`—Requests unsolicited notification service from GIS. See Chapter 7 for details.

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

Table 2 outlines the relationships among the C# examples. Depending on what your client application needs to do, you must implement at least some, if not all, of the functionality exercised in the following dependent levels.

Table 2: Session and Statistics Example Hierarchy

API Feature Examples	Example Filename
Level 1: Create Session	CreateSessionExample.cs
Level 2: Connect Session Service	ConnectSessionServiceExample.cs
Level 3: Browse Services	IdentifyServicesExample.cs
Level 2 + Retrieve Statistic	RetrieveStatisticExample.cs
Level 2 + Retrieve Statistical Profile	RetrieveStatisticalProfileExample.cs
Level 4: Subscribe to Statistics	RetrieveSubscribedStatisticExample.cs
Level 2 + Unsolicited Notification	UnsolicitedNotifExample.cs

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 Statistics SDK Service. You do not need a license to use the Session Service. After you have checked out a Statistics SDK Service license using the Session Service, your client application can send requests to the Statistics SDK Service.

The Session ID

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

Session Example Files

There are three session example files, all of which are located in the `gis.sessExample` package:

- `CreateSessionExample.cs`—Establishes a session with GIS. Used widely in the Statistic 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 8.

About the Statistics Service Examples

To get a statistic, your application has to make a request for or check out a statistic, and then analyze the response.

The statistic examples demonstrate how to subscribe to a statistic, retrieve a statistic, and request unsolicited notification service from a GIS Server. You can use the code examples to learn how to create the dependent or necessary objects to accomplish these tasks.

Statistics Service Example Files

There are four statistic examples. They are:

- `RetrieveStatisticExample.cs`—Retrieves the current value for a statistic. See Chapter 10 for details.
- `RetrieveStatisticalProfileExample.cs`—Retrieves the parameters for a specified statistic. See Chapter 9 for details.
- `SubscribeToStatisticExample.cs`—Subscribes to statistics. See Chapter 11 for details.

- `RetrieveSubscribedStatisticExample.cs`—Retrieves values for statistics that you have subscribed to. See Chapter 11 for details.
- `UnsolicitedNotifExample.cs`—Requests unsolicited notification service from GIS. See Chapter 12 for details.

All the statistic examples are located in the directory 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*. 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.

Your system must use Apache AXIS v. 1.1 or higher. Genesys recommends that you use a released version of Apache AXIS instead of a beta release.

Code stubs or proxies must be generated from the Session and Statistics WSDL files. For convenience, the code example packages include 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 41 for instructions on how to generate them for Java; for Microsoft C#, see “Generating C# Proxy Files” on page 41.

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.
- JRE 1.4 or higher to run the examples.
- 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 42.

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: If you plan to develop in C++ or Visual Basic, Genesys recommends that you use the Microsoft SOAP Toolkit v. 3.0.

Installing the AXIS Toolkit

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

If you plan to develop in 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, 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, v. 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 Statistics services. These code examples provide a model for development of your GIS-based statistics-gathering 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 Statistics 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 Statistic 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 `/usr/GCTI/Java Statistic SDK WS Examples`.

End of procedure

Example Directory Structure

The directory structure for the examples inside your newly-created directory depends on whether you have unpacked code examples using Java, or C#.

Directory Structure for Java Examples

Figure 9 shows the directory structure of the Java examples.

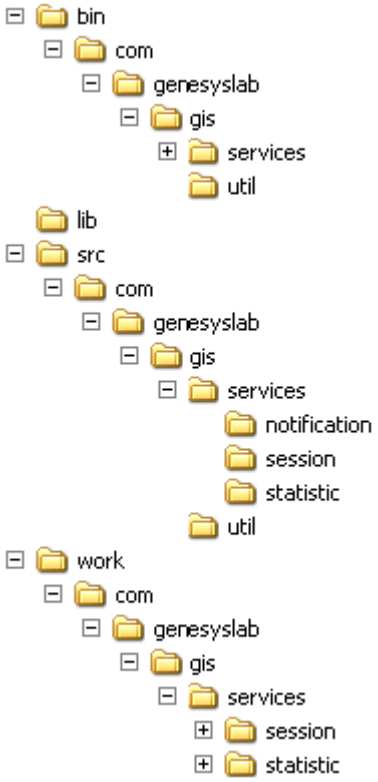

Figure 9: Web Services Java Examples Directory Structure

- `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 used in the examples (Apache Axis Toolkit, ...).
- `src`—This directory contains all of the source files for all examples. It includes the following:
  - `deploy.wsdd`—This file contains regarding services and defines how incoming and outgoing messages will be processed.
  - `com/genesyslab/gis/`—This directory contains the source files for all the examples. It 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`

- `statistic`—This directory contains the following Statistics Service Java source files:
  - `RetrieveStatisticalProfileExample.java`
  - `RetrieveStatisticExample.java`
  - `RetrieveSubscribedStatisticExample.java`
  - `SubscribeToStatisticExample.java`
  - `UnsolicitedNotificationExample.java`
- `util`—This directory contains the following Java files that are used to configure properties and define how services will 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. It includes the following:

Directory Structure for C# Examples

Figure 10 shows the directory structure of the C# examples.

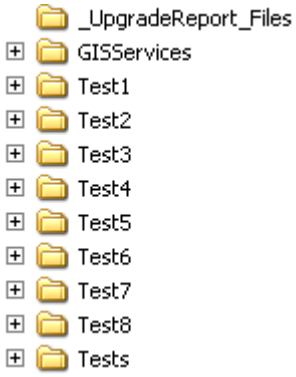

Figure 10: Web Services C# Examples Directory Structure

- `Statistic SDK WS Examples.sln`—The Visual Studio Solution file.
- `GISServices`—This directory contains the Visual Studio Project files for the `GISServices` project. It include the following files needed to run the examples:
  - `GISServices.csproj`
  - `SessionServiceService.cs`
  - `StatServiceService.cs`

  For instructions on using these files or for generating new ones, see “Generating C# Proxy Files” on page 41.
- `Test1`—This directory contains the Visual Studio Project files for the `Test1` project. It include 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. It include 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. It include the following files needed to run the examples:
  - Test4.csproj
  - RetrieveStatisticalProfileExample.cs
- Test5—This directory contains the Visual Studio Project files for the Test5 project. It include the following files needed to run the examples:
  - Test5.csproj
  - RetrieveStatisticExample.cs
- Test6—This directory contains the Visual Studio Project files for the Test6 project. It include the following files needed to run the examples:
  - Test6.csproj
  - SubscribeToStatisticExample.cs
- Test7—This directory contains the Visual Studio Project files for the Test7 project. It include the following files needed to run the examples:
  - Test7.csproj
  - RetrieveSubscribedStatisticExample.cs
- Test8—This directory contains the Visual Studio Project files for the Test8 project. It include the following files needed to run the examples:
  - Test8.csproj
  - UnsolicitedNotificationExample.cs
  - InteropNS.cs
- Tests—This directory contains the Visual Studio Project files for the Tests project. It include 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 Statistics WSDL file defines the messages, operations, SOAP bindings, and elements that make up the GIS interface to License Manager and Stat Server. 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 Statistics 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 *Statistics SDK 7.6 Web Services API Reference* contains complete lists and explanations of all messages and operations included in the Statistics 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 Statistic 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.

Use the WSDL2Java tool at `java org.apache.axis.wsdl.WSDL2Java` to do this. For the Session service, enter:

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

For the Statistics service, enter:

```
-p com.genesyslab.gis.services.statistic http://
<GIS_HOST>:<GIS_PORT>/gis/services/StatService?wsdl
```

Note: These examples assume that you installed GIS on port 8080. If you used a different port number, replace port 8080 with 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, `SessionServiceService.cs` and `StatServiceService.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 .NET SDK.

For the Session service, enter:

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

For the Statistics service, enter:

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

Note: These examples assume that you installed GIS on port 8080. If you used a different port number, replace port 8080 with 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 your location, including connection parameters such as the target host and port number used to connect to GIS.You *must* change these parameters before compiling or running any of the samples.

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 Statistic SDK WS Examples\ant compile
```

Compiling and Running the C# Examples

To run the C# examples, click on `Tests.csproj` to open the `Tests` project in Microsoft Visual Studio. 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>:<gisPort>");
}
```

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

1. Open the Visual Studio Solution File, `Statistics 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. All 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. Table 3 lists and describes the methods included in this example.

Table 3: 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. You will find the stub classes in the `work` folder.

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 in to the GIS Server by configuring an `Identity` object and passing it to the `login` request. If the `login` request is successful, then the GIS 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 Statistic 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. Table 4 lists and describes the methods included in this example.

Table 4: 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 Statistics Service using the `port.getServices()` method. The variable `list_subscribed` contains the list of services checked out. For the Statistic examples, the checked out service will be `GIS_STATSERVICE`.

```
      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 the 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_STATSERVICE`

```
   public static void main(String[] args) {
      String gisHost = PropertiesLoader.getOption("gis.host");
      String gisPort = PropertiesLoader.getOption("gis.port");
      try{
         ConnectSessionServiceExample cs = new ConnectSessionServiceExample(String);
         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 Statistic 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. Table 5 lists and describes the methods included in this example.

Table 5: 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

The Identify Services 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 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 49.

```
   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[i]);
   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 occurred" + serviceException.getMessage());
   }
}
```

Sample Output

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

```
C:\GCTI\Java Statistic 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 Retrieve Statistic Example—Java

This chapter discusses an example of how to use the `RetrieveStatistic` method, and describes which parameters must be defined for the statistic you want to retrieve.

This chapter assumes that you are familiar with Stat Server and how statistics are defined. Refer to the *Framework 7.x Stat Server User's Guide* and the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information.

Note: The Statistics SDK Service does not allow access to any statistics that use the `CurrentState` *statistical category.* This means that you cannot collect real-time agent state statistics. You can, however, collect statistics that have the `AgentStatus` *subject.*

This chapter includes the following sections:

Retrieve Statistic Overview

The Retrieve Statistic example demonstrates how to do a one-time retrieval of a single statistic. You are required to fully define the desired statistic for each request, as described in "Creating Statistic Requests" on page 56.

Retrieving or Subscribing

If you are retrieving the same statistic many times, you may find it more efficient to subscribe to that statistic instead. When you subscribe to a statistic,

GIS returns a `Statistic` object. You can use this object to retrieve updated statistical values, rather than having to specify all of the statistic parameters manually in each request.

For more information about subscribing to statistics, see Chapter 6 on page 71.

Creating Statistic Requests

To request a statistic, you configure and submit a `Statistic` complex type that is composed of the `statisticID`, `objectIDType`, `metric`, and `schedule` types.

- The `StatisticID` is a unique identifier that GIS uses to track the statistic over time. Your client application is responsible for creating a `statisticID` for a particular statistic.
- The `ObjectIDType` describes the particular object for which the statistic is requested. An `objectID` consists of two strings:
  - `ID`—Specifies one of the configuration objects used by Stat Server. For example, to get statistics about an Agent Group, use the name of the Agent Group; to get statistics about a DN, use the format *`<dn name>@<switch name>`*.
  - `tenantName`—Specifies the tenant name, as configured in Configuration Manager.
- A `Metric` specifies parameters for filtering Stat Server data. A `Metric` comprises at least an interval (`timeInterval`) and either a `typeName` string or a `statisticType`.

  The `timeInterval` uses a string to define a window (growing, sliding, sliding selection), and may also specify a `length` integer, `slidelength` integer, or `timeProfileName` depending on the type of window. If a `typeName` is given, then that string must match a name known to Stat Server. If a `statisticType` is included, then the application must supply the data needed for the `statisticType` elements, which resolve to strings that must match their counterparts known to Stat Server.

  A `Metric` may also optionally specify the name of a filter or of a comma-separated `TimeRange` list, as well as the data for a `timeRange` type (two integers: one for left time, the other for right time).
- A `Schedule` directs Stat Server to provide a statistic at a certain interval over time. A `Schedule` has a `timeout` and an `insensitivity` value (both integers), as well as a `notificationMode` that may be `ChangesBased`, `TimeBased`, or `ResetBased`.

Note: Refer to the *Framework 7.x Stat Server User's Guide* for more information about the data required to identify any particular statistic.

Retrieve Statistic Example

The Retrieve Statistic example code is located in the `src\com\genesyslab\gis\services\statistic\RetrieveStatisticExample.java` file. Table 6 lists and describes the methods included in this example.

Table 6: Retrieve Statistic Methods

Method Name	Method Description
`retrieve()`	This method creates a session and logs in, and then creates and submits a statistic request.
`main()`	Use this method to run this class as a stand-alone program.

Retrieve Statistic Example Code

The Retrieve Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

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

import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.services.statistic.types.Metric;
import com.genesyslab.gis.services.statistic.types.ObjectIdType;
import com.genesyslab.gis.services.statistic.types.ObjectType;
import com.genesyslab.gis.services.statistic.types.RetrieveStatisticResponse;
import com.genesyslab.gis.services.statistic.types.Schedule;
import com.genesyslab.gis.services.statistic.types.ScheduleMode;
import com.genesyslab.gis.services.statistic.types.Statistic;
import com.genesyslab.gis.services.statistic.types.StatisticType;
import com.genesyslab.gis.services.statistic.types.StatisticValue;
import com.genesyslab.gis.services.statistic.types.TimeInterval;
import com.genesyslab.gis.services.statistic.types.TimeIntervalType;
import com.genesyslab.gis.util.PropertiesLoader;

public class RetrieveStatisticExample {
   public StatServiceService statService;
   public StatService statPort;
   public RetrieveStatisticExample(){}
```

retrieve() method

The main logic in the code is in the `retrieve()` method.

This method uses the `ConnectSessionServiceExample` to log in and retrieve the Session ID, and then creates a statistic request. The method defines specific information about this statistic most of which are quantitative.

See the *Framework 7.x Stat Server User's Guide* or the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information on statistics configuration.

Note: This example calls the `retrieveStatistic()` method twice. GIS will initialize the statistic when this method is called for the first time. After a 32-second timeout, a second call to this method will retrieve the true value.

```
public void retrieve(String gisServer) throws Exception {
   String[] services = {"GIS_STATSERVICE"};
```

This example uses the `ConnectSessionServiceExample` to log in to GIS, register for a service, and retrieve the Session ID.

```
   ConnectSessionServiceExample cs = new ConnectSessionServiceExample(services);
   cs.execute(gisServer);
```

Next it uses the stub class `StatServiceServiceLocator()` to retrieve a `StatServiceService` object, represented by the `statService` variable. Then through the `StatServiceService` object, it creates a binding to the statistic service URL and session.

```
   // Make a service
   statService = new StatServiceServiceLocator();
   // Now use the service to get a stub which implements the SDI.
   statPort = statService.getStatService(new java.net.URL("http://" + gisServer +
      "/gis/services/StatService?GISsessionId="+ cs.sid));
```

The main section of the code is in building the `Statistic` object. The `subscribeStatistic()` requires the `Statistic` object. Here is the call to initialize a new `Statistic` object.

```
   Statistic statistic = new Statistic();
```

You must provide a `statisticID` if you are subscribing to a statistic. However, in this example, you will not need to reference this `statisticID`.

```
   statistic.setStatisticId("statref1");
```

There are many dependent objects that must be created for the `Statistic` object. The first object being defined is the `ObjectType`.

```
ObjectIdType objectId = new ObjectIdType();
String agentId = PropertiesLoader.getOption("statistic.agentId");
System.out.println("Statistic needed for "+agentId);
```

The object ID identifies a specific object in the configuration object type you specified. In this case, the ID is a specific agent called `Agent1601`.

```
objectId.setId(agentId);
```

The `agent` object is under the Tenant `statistic.tenant`. The `statistic.tenant` is the tenant name defined in the `gis.properties` file.

```
String tenant = PropertiesLoader.getOption("statistic.tenant");
objectId.setTenantName(tenant);
```

After setting the object type, add the object type data to the `Statistic` object.

```
statistic.setObjectId(objectId);
```

Next use the method to create a `Metric` object. The `Metric` object specifies which type of statistical calculation GIS should return. You can include more information than presented in the example.

```
Metric metric = new Metric();
StatisticType statisticType = new StatisticType();
ObjectType[] objectType = new ObjectType[] { ObjectType.Agent };
statisticType.setObjectType(objectType);
metric.setStatisticType(statisticType);
System.out.println("Statistic of interest is agent's total login time");
metric.setTypeName("TotalLoginTime");
TimeInterval timeInterval = new TimeInterval();
timeInterval.setIntervalType(TimeIntervalType.GrowingWindow);
metric.setInterval(timeInterval);
```

Now add the metric data to the `Statistic` object.

```
statistic.setMetric(metric);
```

Finally, the method defines the Schedule information. Schedules specify how frequently you will get statistic updates. Here the schedule mode is `ChangesBased`, which means an update is sent whenever there is a change in the statistic value.

```
Schedule schedule = new Schedule();
schedule.setNotificationMode(ScheduleMode.ChangesBased);
```

Add the schedule data to your `Statistic` object.

```
statistic.setSchedule(schedule);
```

Now that the `Statistic` object is properly set, the code calls the `retrieveStatistic()` method to request a statistic. The method also requires the Stat Server name argument.

Note: The first call using this method for this statistic only initializes the statistic. The value returned is 0. The second call retrieves an actual statistic value.

```
String statServerName = PropertiesLoader.getOption("statistic.statserver.name");
RetrieveStatisticResponse retStat =
        statPort.retrieveStatistic(statistic, statServerName);
System.out.println("Statistic retrieved:");
StatisticValue stat = retStat.getStatisticValue();
System.out.print("Initializing the statistic...");
System.out.println("value retrieved is = "+ stat.getEventValues()[0].getLValue());
```

The example calls the `retrieveStatistic()` again, after a `32` second timeout, to retrieve the statistic.

```
try{
   Thread.sleep(32000);
}catch(InterruptedException interruptedexception) { }
retStat = statPort.retrieveStatistic(statistic, statServerName);
stat = retStat.getStatisticValue();
System.out.print("Retrieving statistic...");
System.out.println("value retrieved is = " +
   stat.getEventValues()[0].getLValue());
}
```

main() method

The code includes a `main()` method for running this class as a stand-alone program.

It creates a new instance of the example and calls the `retrieve()` method, passing in the GIS server host name and port number information as a `String` argument:

```
public static void main(String[] args) {
   String gisHost = PropertiesLoader.getOption("gis.host");
   String gisPort = PropertiesLoader.getOption("gis.port");
   try{
      RetrieveStatisticExample sb = new RetrieveStatisticExample();
```

```
         sb.retrieve(gisHost + ":" + gisPort);
      } catch (Exception serviceException) {
         System.out.println(serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running this example:

```
C:\GCTI\Java Statistic SDK WS Examples\ant "Retrieve Statistic Example"
Buildfile: build.xml

Retrieve Statistic Example:
     [java] - Unable to find required classes (javax.activation.DataHandler
         and javax.mail.internet.
     [java] logging into session server...
     [java] logged into session server successfully.
         Session id =SOAP:SessionService:1221033547948H4
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_STATSERVICE
     [java] Statistic needed for 1000
     [java] Statistic of interest is agent's total login time
     [java] Statistic retrieved:
     [java] Initializing the statistic...value retrieved is = 0
         .... 32 seconds later
     [java] Retrieving statistic...value retrieved is = 32

BUILD SUCCESSFUL
```

Chapter

5 Retrieve Statistical Profile Example—Java

This chapter examines the Retrieve Statistical Profile example.

Statistical profiles are settings that you have defined (using Configuration Manager) in the Stat Server Application object, and contain information such as the time range, time profile, and statistic names available for subscription.

See "Creating Statistic Requests" on page 56 for a short explanation of how to configure a statistic request.

Note: See the *Framework 7.x Stat Server User's Guide* and the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information on statistics configuration.

This chapter includes the following section:

Retrieve Statistical Profile Overview

This example demonstrates how to create session request to retrieve statistical profiles. The response that is returned includes statistic names available for subscription, and any `timerange` and `timeprofile` settings that are available.

Retrieve Statistical Profile Example

The Retrieve Statistical Profile example code is located in the `src\com\genesyslab\gis\services\statistic\RetrieveStatisticalProfileExa`

mple.java file. Table 7 lists and describes the methods included in this example.

Table 7: Retrieve Statistical Profile Methods

Method Name	Method Description
execute()	This method uses the ConnectSessionServiceExample class to log in and check out the license for the statistic service. Then it retrieves the statistical profiles to display available statistic names, time ranges, and time profiles.
main()	Use this method to run this class as a stand-alone program.

Retrieve Statistical Profile Example Code

The Retrieve Statistical Profile example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
package com.genesyslab.gis.services.statistic;
import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.services.statistic.types.ObjectType;
import com.genesyslab.gis.services.statistic.types.Parameter;
import com.genesyslab.gis.services.statistic.types.ProfileInfo;
import com.genesyslab.gis.services.statistic.types.RetrieveStatisticalProfileResponse;
import com.genesyslab.gis.services.statistic.types.StatisticTypeInfoType;
import com.genesyslab.gis.services.statistic.types.StatisticalProfileType;
import com.genesyslab.gis.services.statistic.types.TimeProfile;
import com.genesyslab.gis.util.PropertiesLoader;

public class RetrieveStatisticalProfileExample {
   public RetrieveStatisticalProfileExample() {}
```

execute() Method

The main logic in the RetrieveStatisticalProfile example is contained in the execute() method.

```
   public void execute(String targetHost) throws Exception {
      StatService portStat;
      String statEndPoint = "";
      String[] services = {"GIS_STATSERVICE"};
```

The `ConnectSessionServiceExample` is used to connect to GIS and register for the statistics service. The Session ID can also be referenced from this class.

```
ConnectSessionServiceExample cs = new ConnectSessionServiceExample(services);
cs.execute(targetHost);
```

Using the Session ID from the `ConnectSessionServiceExample` object, you can bind to a statistic service using the `StatServiceServiceLocator` stub class. After binding the service, you can send statistic requests to GIS.

```
StatServiceServiceLocator service_stat = new StatServiceServiceLocator();
portStat = service_stat.getStatService();
statEndPoint = "http://" + targetHost + "/gis/services/StatService?GISsessionId="
+ cs.sid;
//Bind stat address endpoint
((StatServiceSoapBindingStub)portStat)
   ._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,statEndPoint);
```

The next step is to use the `SessionService` object to request the statistical profile type. The response for this request contains statistic type data that can be displayed using the `getTypeName()` and `getValue()` methods.

```
RetrieveStatisticalProfileResponse statistic_profile =
   portStat.retrieveStatisticalProfile("",
      StatisticalProfileType.statisticalProfile);
ProfileInfo info = statistic_profile.getStatisticalProfileInfo();
StatisticTypeInfoType[] types = info.getStatisticInfos();
for (int i = 0; i < types.length; i++) {
   StatisticTypeInfoType infoType = types[i];
   System.out.print(infoType.getTypeName() + "( " +
      infoType.getType().getValue() + ") ->");
   for (int j = 0; j < infoType.getObjectTypes().length; j++) {
      ObjectType type = infoType.getObjectTypes()[j];
      System.out.print(": " + type.getValue());
   }
   System.out.println("\n----");
}
```

Using similar code as above, a request is sent to retrieve the filter profiles for the statistic. Results are displayed using the `getKey()` method to retrieve the filter name and the `getValue()` method to return the filter value.

```
RetrieveStatisticalProfileResponse filterProfile =
   portStat.retrieveStatisticalProfile("",
   StatisticalProfileType.filterProfile);
ProfileInfo info1 = filterProfile.getStatisticalProfileInfo();
Parameter[] filter = info1.getFilters();
if (filter != null)
   for (int i =0;i<filter.length; i++)
```

```
        System.out.println("Key= " + filter[i].getKey() + "  Value= " +
          filter[i].getValue());
```

Request the time profile for the statistic, and then use the `getKey()` and `getValue()` methods to display the name of the filter and the value of the time interval, respectively.

```
  RetrieveStatisticalProfileResponse time_profile =
    portStat.retrieveStatisticalProfile("",
    StatisticalProfileType.timeProfile);
  ProfileInfo info2 = time_profile.getStatisticalProfileInfo();
  TimeProfile[] timeProf = info2.getTimeProfiles();
  if (timeProf != null)
    for (int i =0;i<timeProf.length; i++){
      System.out.print("Key = " + timeProf[i].getKey() + "  Value = " +
        timeProf[i].getValue());
  System.out.println("  time interval = " +
    timeProf[i].getIntervalType().getValue());
  }
```

One last request is created to retrieve time range profiles. The `getKey()` method displays the name of the filter and `getValue()` method displays the value of the time range.

```
  RetrieveStatisticalProfileResponse timeRangeProfile =
    portStat.retrieveStatisticalProfile("",
    StatisticalProfileType.timeRangeProfile);
  ProfileInfo info3 = time_profile.getStatisticalProfileInfo();
  Parameter[] timeRange = info3.getTimeRanges();
  if (timeRange != null)
    for (int i =0;i<timeRange.length; i++)
      System.out.println("Key = " + timeRange[i].getKey() + "  Value = " +
        timeRange[i].getValue());
```

The method also logs out of the session using the `SessionService` object:

```
  cs.port.logout(cs.sid);
}
```

main() Method

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

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

```
      try {
         RetrieveStatisticalProfileExample example = new
            RetrieveStatisticalProfileExample();
         example.execute(gisHost+":"+gisPort);
      } catch (Exception serviceException) {
         System.out.println("error occurred: " + serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running this example:

```
C:\GCTI\Java Statistic SDK WS Examples\ant "Retrieve Statistical Profile Example"

Retrieve Statistical Profile Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221034817627H5
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_STATSERVICE
     [java] retrieving statistic profiles.....
     [java] ServiceFactor( fValue) ->: Queue: RoutePoint: GroupQueues
     [java] ----
     [java] CurrentPlaceState( unknownValue) ->: Place
     [java] ----
     [java] AverOutboundStatusTime( fValue) ->: Agent: Place: GroupAgents: GroupPlaces
     [java] ----
     [java] TotalNumberCallsAband( lValue) ->: Queue: RoutePoint: GroupQueues
     [java] ----
     [java] TotalNumberInboundCalls( lValue) ->: RegDN: Agent: Place:
        GroupAgents: GroupPlaces
     [java] ----
     [java] TotalNotReadyStatusTime( lValue) ->: Agent: Place: GroupAgents: GroupPlaces
     [java] ----
     [java] TotalTalkDNStatusTime( lValue) ->: RegDN: Agent: Place:
        GroupAgents: GroupPlaces
     [java] ----
     [java] CurrentDNState( unknownValue) ->: RegDN
     [java] ----
     [java] CurrNumberNotReadyStatuses( lValue) ->: GroupAgents: GroupPlaces
     [java] ----
     [java] AverInboundStatusTime( fValue) ->: Agent: Place: GroupAgents: GroupPlaces
     [java] ----
     [java] CurrNumberOutboundStatuses( lValue) ->: GroupAgents: GroupPlaces
     [java] ----
     [java] CurrentGroupState( unknownValue) ->: GroupAgents: GroupPlaces
     [java] ----
```

```
[java] AverConsultStatusTime( fValue) ->: Agent: Place: GroupAgents: GroupPlaces
[java] ----
[java] TotalNumberOutboundCalls( lValue) ->: RegDN: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] AverInboundPlaceStatusTime( fValue) ->: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] CurrNumberRingingStatuses( lValue) ->: GroupAgents:
   GroupPlaces
[java] ----
[java] TotalAfterCallWorkStatusTime( lValue) ->: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] TotalNotReadyPlaceStatusTime( lValue) ->: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] TotalAfterCallWorkPlaceStatusTime( lValue) ->: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] CurrNumberConsultStatuses( lValue) ->: GroupAgents: GroupPlaces
[java] ----
[java] AbandCallsPercentage( fValue) ->: Queue: RoutePoint: GroupQueues
[java] ----
[java] CurrentAgentState( unknownValue) ->: Agent
[java] ----
[java] TotalNumberConsultCalls( lValue) ->: RegDN: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] AverAbandCallTime( fValue) ->: Queue: RoutePoint: GroupQueues
[java] ----
[java] AverHandleDNStatusTime( fValue) ->: RegDN: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] EstimTimeToDistribCall( fValue) ->: Queue: RoutePoint: GroupQueues
[java] ----
[java] CurrNumberInternalStatuses( lValue) ->: GroupAgents: GroupPlaces
[java] ----
[java] CurrNumberWaitingCalls( lValue) ->: Queue: RoutePoint: GroupQueues
[java] ----
[java] CurrNumberInboundStatuses( lValue) ->: GroupAgents: GroupPlaces
[java] ----
[java] AverInboundDNStatusTime( fValue) ->: RegDN: Agent: Place:
   GroupAgents: GroupPlaces
[java] ----
[java] AverHandleStatusTime( fValue) ->: Agent: Place: GroupAgents: GroupPlaces
[java] ----
[java] CurrNumberWaitStatuses( lValue) ->: GroupAgents: GroupPlaces
[java] ----
[java] AverOutboundPlaceStatusTime( fValue) ->: Agent: Place:
   GroupAgents: GroupPlaces
```

```
     [java] ----
     [java] TotalTalkPlaceStatusTime( lValue) ->: Agent: Place: GroupAgents: GroupPlaces
     [java] ----
     [java] CurrNumberHoldStatuses( lValue) ->: GroupAgents: GroupPlaces
     [java] ----
     [java] TotalNumberInternalCalls( lValue) ->: RegDN: Agent: Place:
        GroupAgents: GroupPlaces
     [java] ----
     .....
     [java] TotalAfterCallWorkDNStatusTime( lValue) ->: RegDN: Agent:
        Place: GroupAgents: GroupPlaces
     [java] ----
     [java] TotalTalkStatusTime( lValue) ->: Agent: Place: GroupAgents: GroupPlaces
     [java] ----
     ..............
     [java] Key = SlidingDay  Value = 86400:3600  time interval = SlidingWindow
     [java] Key = SlidingHour  Value = 3600:600  time interval = SlidingWindow
     [java] Key = Default  Value = 0:00  time interval = GrowingWindow
     [java] retrieving time range profiles.....
     [java] logging out of session...

BUILD SUCCESSFUL
```

Chapter

6 Subscribe and Retrieve Examples—Java

This chapter examines the subscribe to/retrieve statistics examples.This chapter includes the following sections:

Subscribing to Statistics

Rather than creating and submitting multiple individual requests to track changes in the value of a statistic over time, you can subscribe to an available statistic. This provides a `Statistic` object that allows you to check for statistical updates quicker and easier than if you had to specify all of the statistic parameters for each request.

Creating a Subscription

To create a subscription to a statistic, you specify all of the parameters required to build a subscription object and then use the `subscribeStatistic()` method to send a subscription request to the GIS statistics session.

Once the subscription has been created, you can reference it using a `subscription` object with the following properties:

- `statisticID`—The unique identifier for a statistic, assigned by you when created the subscription.

- `scope`—Corresponds to the number of values for each particular `statisticID`, calculated between one time the application subscribes and the next. The application can retrieve a set of last values for both subscription instances and use them as the scope.

To remove a subscription, use the unique `statisticID` with the `subunsubscribeStatistic` request.

Retrieving the Value of a Subscribed Statistic

To retrieve the value of a subscribed statistic, you send a list of statistic subscriptions and a `Notification` object to the `retrieveSubscribedStatistics` request.

Note: It is possible to retrieve one-to-many subscribed statistics with one SOAP request. For details, see the *Statistics SDK 7.6 Web Services API Reference* > Chapter 7:Statistics SDK Service Operations.

The `retrieveSubscribedStatistics` request is limited by a restriction time limit. Your client application must wait for the period (in seconds) set by the `restriction_time` option in the GIS Application object `Options` tab before sending another retrieve request. You can adjust the length of this setting from Configuration Manager, as described in the *Genesys Integration Server 7.6 Deployment Guide*.

Subscription Notification Types

If your statistic subscription request `unsolicitedNotification` type includes the URL for your client application's web server as its value, GIS sends unsolicited notification whenever the data for your subscribed statistics is updated. For additional details about unsolicited notification, see Chapter 7 on page 81.

If the `unsolicitedNotification` type is null, then you must set the notification mode to either:

- `Polling`
- `Blocked`

For details about notification modes, see Chapter 1, "Notification Modes" on page 21.

Overview of Examples

This chapter discusses the following examples:

- Subscribe To Statistic Example—This example demonstrates how to subscribe and unsubscribe to a statistic. Most of the code for this example is required to build a `statistic` object used in the subscribe request.

- Retrieve Subscribed Statistic Example—This example demonstrates how to retrieve a subscribed statistic using the Polling mode of solicited notification. The example uses the Subscribe To Statistic example to subscribe and unsubscribe to the statistic.

  In this example, the object of interest is an agent and the statistic retrieved is the total login time for that agent. We are also assuming the agent object is under the Tenant Resources. This example can be modified slightly to retrieve statistics for Agents, Queues, Places, or other object types.

Subscribe To Statistic Example

The Subscribe To Statistic example code is located in the `src\com\genesyslab\gis\services\statistic\SubscribeToStatisticExample.java` file. Table 8 lists and describes the methods included in this example.

Table 8: Subscribe To Statistic Methods

Method Name	Method Description
`subscribe()`	This method creates a session, logs into a GIS server, and subscribes to a statistic.
`unsubscribe()`	This method is used to unsubscribe from the statistic created in the `subscribe()` method.
`main()`	Use this method to run this class as a stand-alone program.

Subscribe To Statistic Example Code

The Subscribe To Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

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

import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.services.statistic.types.Metric;
import com.genesyslab.gis.services.statistic.types.ObjectIdType;
import com.genesyslab.gis.services.statistic.types.ObjectType;
import com.genesyslab.gis.services.statistic.types.Schedule;
import com.genesyslab.gis.services.statistic.types.ScheduleMode;
import com.genesyslab.gis.services.statistic.types.Statistic;
import com.genesyslab.gis.services.statistic.types.StatisticType;
import com.genesyslab.gis.services.statistic.types.TimeInterval;
import com.genesyslab.gis.services.statistic.types.TimeIntervalType;
```

```
import com.genesyslab.gis.services.statistic.types.UnsolicitedNotification;
import com.genesyslab.gis.util.PropertiesLoader;

public class SubscribeToStatisticExample{
   StatServiceService statService;
   StatService statPort;
   public SubscribeToStatisticExample(){}
```

subscribe() Method

The `subscribe()` method accomplishes all the application logic for the example. This method uses the `ConnectsessionServiceExample` to log in and retrieve the Session ID, and then subscribes to a statistic. The statistic subscribed in this example is for an agent's total login time.

```
   public void subscribe(String gisServer) throws Exception {
      String[] services = {"GIS_STATSERVICE"};
```

First, it uses the `ConnectSessionoServiceExample` to log in to GIS, register for a service, and retrieve the Session ID.

```
      ConnectSessionServiceExample cs = new ConnectSessionServiceExample(services);
      cs.execute(gisServer);
```

Next, it uses the stub class `StatServiceServiceLocator()` to retrieve a `StatServiceService` object, represented by the `statService` variable. Then through the `StatServiceService` object, it creates a binding to the Statistic service URL and session.

```
      statService = new StatServiceServiceLocator();
      statPort = statService.getStatService(new java.net.URL("http://" + gisServer + "/
         gis/services/StatService?GISsessionId="+ cs.sid));
```

Much of the code in this method is used to build the `Statistic` object required by the `subscribeStatistic()` method. Here is the call to initialize a new `Statistic` object.

```
      Statistic statistic = new Statistic();
```

You must provide a `statisticID` if you are subscribing to a statistic. GIS uses this `statisticID` to send subsequent statistic updates for the current session.

```
      statistic.setStatisticId("statref1");
```

There are many dependent objects created for the `Statistic` object. The first object being defined is the `ObjectType`.

```
ObjectIdType objectId = new ObjectIdType();
```

A unique ID identifies a specific object in the configuration object type you specified. Here the ID is a specific agent called `Agent1601`.

```
String agentId = PropertiesLoader.getOption("statistic.agentId");
System.out.println("Statistic needed for " + agentId);
objectId.setId(agentId);
```

The next line says that the agent object is under the Tenant `Resources`.

```
String tenant = PropertiesLoader.getOption("statistic.tenant");
objectId.setTenantName(tenant);
```

After setting the object type, the code adds the object type data to the `Statistic` object.

```
statistic.setObjectId(objectId);
```

Next, the method creates a `Metric` object, which defines the type of statistical calculation GIS should return.

```
Metric metric = Metric();
StatisticType statisticType = new StatisticType();
```

The object type identifies which type of configuration object you are interested in. In this code, the object type of interest is `Agent`, so the code sets the `ObjectType` with `ObjectType.Agent`.

```
ObjectType[] objectType = new objectType[]{ObjectType.Agent};
statisticType.setObjectType(objectType);
```

Using the Metric object created earlier, you specify the type and interval of statistics to retrieve. You can specify even more information than listed in this example.

```
metric.setStatisticType(statisticType);
System.out.println("Statistic interested is agent's total login time");
metric.setTypeName("TotalLoginTime");
TimeInterval timeInterval = new TimeInterval();
timeInterval.setIntervalType(TimeIntervalType.GrowingWindow);
metric.setInterval(timeInterval);
```

Now add the metric data to the `Statistic` object.

```
statistic.setMetric(metric);
```

Finally, the method defines the Schedule information. Schedules specify how often you will get statistic updates. In this case, the schedule mode is `ChangesBased`. Your client will get an update only when there is a change in statistic value.

```
    Schedule schedule = new Schedule();
    schedule.setNotificationMode(ScheduleMode.ChangesBased);
```

This line adds the schedule data to your `Statistic` object.

```
    statistic.setSchedule(schedule);
```

Now that the `Statistic` object is properly set, the code calls the `subscribeStatistic()` method to request a subscription to this statistic. The method also requires two other arguments: the Stat Server name and an `unsolicitedNotification` object. (The `unsolicitedNotification` object can be `null`, as in this example.)

If the `subscribeStatistic` request `unsolicitedNotification` type has a `null` value, you must specify the notification mode you want to use to retrieve the statistic in the request. The default mode is `Polling`.

See Chapter 7 for more information regarding this argument option.

```
    String statServeName = PropertiesLoader.getOption("statistic.statserver.name");
    statPort.subscribeStatistic(statistic, statServeName,
                                    new UnsolicitedNotification());
    System.out.println("Sucessfully subscribed to statistic.");
}
```

unsubscribe() Method

This method is used to unsubscribe from the `statref1` statistic created in the `subscribe()` method.

```
  public void unsubscribe()throws Exception{
    statPort.unsubscribeStatistic("statref1");
    System.out.println("Unsubscribed to statistics.");
  }
```

main() Method

This is the stand-alone option you can use to test-run this example. It creates a new instance of the 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{
         SubscribeToStatisticExample sb = new SubscribeToStatisticExample();
         sb.subscribe(gisHost + ":" + gisPort);
         sb.unsubscribe();
      } catch (Exception serviceException) {
         System.out.println(serviceException.getMessage());
      }
   }
}
```

Sample Output

The following sample output shows one possible result when running this example:

```
C:\GCTI\Java Statistic SDK WS Examples\ant "Subscribe To Statistic Example"
Buildfile: build.xml

Subscribe To Statistic Example:
     [java] logging into session server...
     [java] logged into session server successfully.
        Session id =SOAP:SessionService:1221035280780H7
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_STATSERVICE
     [java] Statistic needed for 1000
     [java] Statistic interested is agent's total login time
     [java] Successfully subscribed to statistic.
     [java] Unsubscribed to statistics.

BUILD SUCCESSFUL
```

Retrieve Subscribed Statistic Example

The Retrieve Subscribed Statistic example code is located in the `src\com\genesyslab\gis\services\statistic\RetrieveSubscribedStatisticExample.java` file. Table 9 on page 78 lists and describes the methods included in this example.

Table 9: Retrieve Subscribed Statistic Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then retrieves the subscribed statistic using the Polling notification mode.
`main()`	Use this method to run this class as a stand-alone program.

Retrieve Subscribed Statistic Example Code

The Retrieve Subscribed Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

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

import com.genesyslab.gis.services.statistic.types.EventValue;
import com.genesyslab.gis.services.statistic.types.Notification;
import com.genesyslab.gis.services.statistic.types.NotificationMode;
import com.genesyslab.gis.services.statistic.
                                        types.RetrieveSubscribedStatisticsResponse;
import com.genesyslab.gis.services.statistic.types.StatisticSubscription;
import com.genesyslab.gis.services.statistic.types.StatisticSubscriptions;
import com.genesyslab.gis.services.statistic.types.StatisticValue;
import com.genesyslab.gis.util.PropertiesLoader;
```

execute() Method

The main logic in the example resides in the `execute()` method. This method retrieves the subscribed statistic using the `Polling` notification mode. Below is an overview of the method.

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

The method first makes a call to `SubscribeToStatistic` example to establish a session and then subscribe to the statistic `Total_Login_Time`:

```
  SubscribeToStatisticExample sb = new SubscribeToStatisticExample();
  sb.subscribe(gisServer);
```

After a client made a subscription request with a fully populated `Statistic` object, GIS can retrieve a statistic through a `StatisticSubscription` object.

The client only needs to set the `statisticID` property with the `statisticID` specified during subscription. The following five lines create and set the `statisticID` and scope of a `Subscription` object:

```
StatisticSubscriptions subscriptions = new StatisticSubscriptions();
StatisticSubscription subscription = new StatisticSubscription();
subscription.setScope("all");
subscription.setStatisticId("statref1");
subscriptions.setStatisticSubscription(new StatisticSubscription[]{subscription});
```

Now create a `Notification` object and set the notification mode and timeout. The timeout is in seconds:

```
Notification notification = new Notification();
notification.setMode(NotificationMode.Polling);
notification.setTimeout(new java.math.BigInteger("10"));
```

Finally, pass the subscription and notification objects to the `retrieveSubscribedStatistics()` method:

```
RetrieveSubscribedStatisticsResponse response =
   sb.statPort.retrieveSubscribedStatistics(subscriptions, notification);
//returns the value
System.out.println("Retrieved statistic value = "+ ((EventValue)((StatisticValue)
   response.getStatisticValues()[0]).getEventValues()[0]).getLValue());
}
```

main() Method

The code includes a `main()` method for running this class as a stand-alone program. It creates a new instance of the example and calls the `execute()` method, passing in the GIS server host name and port number information as a `String` argument.

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

Sample Output

The following sample output shows one possible result when running this example:

```
C:\GCTI\Java Statistic SDK WS Examples\ant "Retrieve Subscribed Statistic Example"
Retrieve Subscribed Statistic Example:
     [java] logging into session server...
     [java] logged into session server Successfully. Session id
   =SOAP:SessionService:1221035360530H8
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_STATSERVICE
     [java] Statistic needed for 1000
     [java] Statistic interested is agent's total login time
     [java] Successfully subscribed to statistic.
     [java] Retrieved statistic value = 45

BUILD SUCCESSFUL
```

Chapter

7 Unsolicited Notification Example—Java

This chapter examines the Unsolicited Notification example. This chapter includes the following sections:

About Unsolicited Notification

GIS provides an unsolicited notification service feature for the Statistics SDK Service. This feature allows GIS to transmit updates to subscribed statistics to a designated HTTP server on the client-side that uses a notification service to distribute the updated information to client applications.

The examples included in this release provide an example of one way you might implement the Unsolicited Notification feature. They show how a client application can draw updated statistical information from your client-side HTTP server using the following steps:

1. A client application sends a statistic subscription request, together with the URL of a client-side HTTP server, to GIS.
2. Whenever the subscribed statistic is updated, GIS sends notification of the change to the client-side HTTP server.
3. The client-side HTTP server saves the received data into cache and sends updates to the Tomcat console.
4. Client applications receive updated statistical information from a service provided by your client-side HTTP server.

Note: The client application receiving updates from the client-side HTTP server does not have to be the same application that created the subscription request, as long as the subscription ID is known.

Figure 11 shows the architecture used in the unsolicited notification examples.

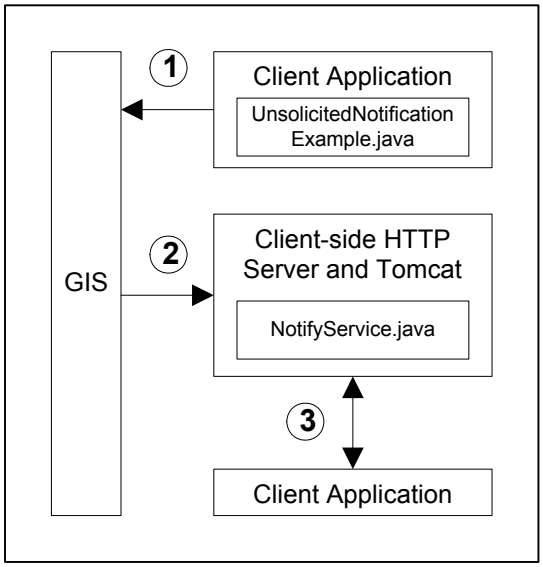

Figure 11: Architecture for Unsolicited Notification

Therefore, this section contains two sets of example code:

- Code used by your client application to send GIS a subscription request and set up unsolicited notification.

  See "Unsolicited Notification Example" on page 84.

- Code implemented in your client-side HTTP server so it can receive notification messages from GIS.

  See "Notification Service Overview" on page 88.

This is only one possible method that you might use to implement unsolicited notification. The only requirements for using unsolicited notification are:

- An HTTP-capable server to receive notification updates.
- Implementation of the notification interface on the HTTP server.

Note: Genesys recommends that you be familiar with developing web services before attempting to implement unsolicited notification.

Unsolicited Event Structure

Once you subscribe to the statistic, the client starts to receive notification events from GIS. A sample notification event (HTTP Request) that could be sent by GIS is included below for your reference:

```
POST /axis/services/notification HTTP/1.1 Content-Type: text/xml; charset="utf-8"
   SOAPAction: notify User-Agent: Java1.3.1_01 Host: <client_host>:<remoting_port>
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2 Connection: keep-alive
   Content-length: 1367

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:tns="http://www.genesyslab.com/notification"
     xmlns:types="http://www.genesyslab.com/notification/encodedTypes"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <tns:notify>
      <eventType xsi:type="xsd:string">StatisticNotification</eventType>
      <keyVal href="#id1" />
    </tns:notify>
    <soapenc:Array id="id1" soapenc:arrayType="xsd:string[][5]">
      <Item href="#id2" />
      <Item href="#id3" />
      <Item href="#id4" />
      <Item href="#id5" />
      <Item href="#id6" />
    </soapenc:Array>
    <soapenc:Array id="id2" soapenc:arrayType="xsd:string[2]">
      <Item>sessionId</Item>
      <Item>SessionService:1040391220078H5</Item>
    </soapenc:Array>
    <soapenc:Array id="id3" soapenc:arrayType="xsd:string[2]">
      <Item>statId</Item>
      <Item>statref1</Item>
    </soapenc:Array>
    <soapenc:Array id="id4" soapenc:arrayType="xsd:string[2]">
      <Item>date</Item>
      <Item>1040391882</Item>
    </soapenc:Array>
    <soapenc:Array id="id5" soapenc:arrayType="xsd:string[2]">
      <Item>intervalLength</Item><Item>386</Item>
    </soapenc:Array>
    <soapenc:Array id="id6" soapenc:arrayType="xsd:string[2]">
      <Item>lValue</Item>
      <Item>366</Item>
    </soapenc:Array>
  </soap:Body>
</soap:Envelope>
```

This example request is composed of a `StatisticNotification` notification type and a `Keyval` array that contains the following values:

- `sessionId`—Client `SessionId` to which the current statistic belongs.

Note: The client-side code should filter the notification messages based on the current `sessionId` to filter out previously-cached events.

- `statId`—The unique `statisticID` value for the current statistic.
- `date`—Date when the statistic value was issued from Stat Server.
- `intervalLength`—Length of the interval that is received from Stat Server.
- `lvalue`, `fvalue`—The statistic value, as calculated by Stat Server.

Note: Each time a notification is successfully sent to a client, that client's `sessionTimeout` timer is rescheduled. Once a notification attempt fails (because the value specified by the `maxAttempts` parameter is exceeded), the `sessionTimeout` is no longer rescheduled.

You can set the `maxAttempts` value by editing your `modules.conf` configuration file. For additional details about this setting, see the *Genesys Integration Server 7.6 Deployment Guide*.

Unsolicited Notification Overview

This example, `src\com\genesyslab\gis\services\statistic\UnsolicitedNotificationExample.java`, demonstrates how to request an unsolicited notification service from GIS. The code in this method is almost identical to `SubscribeToStatistic.java`. The difference is that in `SubscribeToStatistic.java`, the `UnsolicitedNotification` object is left null instead of being populated.

In this example, your client HTTP server's host and port information is set in the `UnsolicitedNotification` object and passed along with the statistic subscription request. Including this information triggers GIS to send statistics updates to the specified URL.

Unsolicited Notification Example

The Unsolicited Notification example code is located in the `src\com\genesyslab\gis\services\statistic\UnsolicitedNotificationExample.java` file. Table 10 lists and describes the methods included in this example.

Table 10: Unsolicited Notification Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then requests an unsolicited notification server from GIS.
`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 unusual processes.

```
package com.genesyslab.gis.services.statistic;
import com.genesyslab.gis.services.session.ConnectSessionServiceExample;
import com.genesyslab.gis.services.statistic.types.Metric;
import com.genesyslab.gis.services.statistic.types.ObjectIdType;
import com.genesyslab.gis.services.statistic.types.ObjectType;
import com.genesyslab.gis.services.statistic.types.Schedule;
import com.genesyslab.gis.services.statistic.types.ScheduleMode;
import com.genesyslab.gis.services.statistic.types.Statistic;
import com.genesyslab.gis.services.statistic.types.StatisticType;
import com.genesyslab.gis.services.statistic.types.TimeInterval;
import com.genesyslab.gis.services.statistic.types.TimeIntervalType;
import com.genesyslab.gis.services.statistic.types.UnsolicitedNotification;
import com.genesyslab.gis.util.NotificationModule;
import com.genesyslab.gis.util.PropertiesLoader;

public class UnsolicitedNotificationExample {
   StatService statPort;
   ConnectSessionServiceExample cs;
   public UnsolicitedNotificationExample() {}
```

execute() Method

The `UnsolicitedNotification` example's main logic is in the `execute()` method.

```
   public void execute(String gisServer) throws Exception {
      String[] services = {"GIS_STATSERVICE"};
```

This section is the same as the `execute()` method in `SubscribeToStatistic.java`. See the explanation for "Subscribe To Statistic Example" on page 73.

```
cs = new ConnectSessionServiceExample(services);
cs.execute(gisServer);
```

First, create a service.

```
StatServiceServiceLocator statService = new StatServiceServiceLocator();
```

Then use the service to get a stub which implements the Session ID.

```
statPort = statService.getStatService(new java.net.URL("http://" + gisServer +
   "/gis/services/StatService?GISsessionId="+ cs.sid));
```

Next build a statistic object, a process that involves a number of steps.

```
Statistic statistic = new Statistic();
statistic.setStatisticId("statref1");
```

Build the `Object ID`.

```
ObjectIdType objectId = new ObjectIdType();
String agentId = PropertiesLoader.getOption("statistic.agentId");
System.out.println("Statistic needed for " + agentId);
```

Enter the ID of the employee whose statistics you are interested in. For this example, the Agent object is under the Tenant `Resources`.

```
objectId.setId("agentId");
```

Add the `Object ID` data to the statistic.

```
packString tenant = PropertiesLoader.getOption("statistic.tenant");
objectId.setTenantName(tenant);
statistic.setObjectId(objectId);
```

Then build metric data and add it to the statistic.

```
Metric metric = new Metric();
StatisticType statisticType = new StatisticType();
ObjectType[] objectType = new ObjectType[]{ObjectType.Agent};
statisticType.setObjectType(objectType);
metric.setStatisticType(statisticType);
System.out.println("Statistic of interest is the agent's total login time");
metric.setTypeName("TotalLoginTime");
```

```
TimeInterval timeInterval = new TimeInterval();
timeInterval.setIntervalType(TimeIntervalType.GrowingWindow);
metric.setInterval(timeInterval);
statistic.setMetric(metric);
```

Specify the statistic update schedule. You can choose from several schedule possibilities, such as `ChangesBased` and `ResetBased`.

```
Schedule schedule = new Schedule();
schedule.setNotificationMode(ScheduleMode.ChangesBased);
statistic.setSchedule(schedule);
```

Now create the `unsolicitedNotification` object and passes in the URL of a client-side notification service.

Note: The client-side server must implement the service according to the GIS `notification.wsdl` specification.

```
String gisHost = PropertiesLoader.getOption("gis.host");
String notificationPort = PropertiesLoader.getOption("client.notification.port");
UnsolicitedNotification un = new UnsolicitedNotification();
un.setUrl("http://" + gisHost + ":" + notificationPort +
   "/axis/services/notification");
//this url points to your client http server's notification service
```

Next, subscribe to the statistic. The `statistic.statserver.name` is the application name of a particular stat server defined in the `gis.properties` file:

```
String statServeName = PropertiesLoader.getOption("statistic.statserver.name");
NotificationModule.getInstance().start();
```

The `subscribe` request using the UnsolicitedNotification mode.

```
statPort.subscribeStatistic(statistic, statServeName, un);
```

main() Method

The code includes a `main()` method for running this class as a stand-alone program. It creates a new instance of the example and calls the `execute()` method, passing in the GIS server's host name and GIS server's port number information as a `String` argument.

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

```
        UnsolicitedNotificationExample example = new UnsolicitedNotificationExample();
        example.execute(gisHost + ":" + gisPort);
    } catch (Exception serviceException) {
        System.out.println(serviceException.getMessage());
    }
```

Notification Service Overview

The Notification Service example code demonstrates possible code for your client-side server. This example include code to create a notification service to receive unsolicited updates from GIS, and a method that allows remote agent desktop clients to access these GIS published data.

Publishing the Notification Service

Before your client-side server can receive unsolicited notification data, you must publish the `notify()` method as a service. The service then needs to be deployed using the `deploy.wsdd` file. The `src\com\genesyslab\gis\util\NotificationModule.java` file deploys/starts and stops an implementation of the client's HTTP Server.

Notification Module Example Code

The Notification Module example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
package com.genesyslab.gis.util;

import java.net.ServerSocket;
import org.apache.axis.configuration.FileProvider;
import org.apache.axis.transport.http.SimpleAxisServer;

public class NotificationModule {
```

Axis Implementation of the HTTP Server.

```
  private SimpleAxisServer notificationServer;
  private static NotificationModule instance;

  private NotificationModule() {
```

First, retrieve the client HTTP server notification port.

```
    int notificationPort = Integer.parseInt(PropertiesLoader.
      getOption("client.notification.port"));
```

```
      notificationServer = new SimpleAxisServer();
      try {
         notificationServer.setServerSocket(new ServerSocket(notificationPort));
```

Next, read the `wsdd` file to get information on how to deploy the Notification Service.

```
         FileProvider configProvider = new FileProvider(this.getClass().
            getClassLoader().getResourceAsStream("deploy.wsdd"));
         notificationServer.setMyConfig(configProvider);
      }catch (Exception e) {
         e.printStackTrace();
      }
   }

   public static NotificationModule getInstance() {
      if (instance == null)
         instance = new NotificationModule();
      return instance;
   }
```

Start the server on client side.

```
   public void start() {
      try {
         notificationServer.start();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

Stop the server on client side.

```
   public void stop() {
      notificationServer.stop();
   }
}
```

Notification Service Example

The Notification Service example code is located in the `src\com\genesyslab\gis\services\notification\NotificationService.java` file. Table 11 lists and describes the methods included in this example.

Table 11: Notify Service Methods

Method Name	Method Description
`notify()`	Accepts notification of statistic or configuration requests, and stores the data into a static hashtable.
`retrieveMessage()`	Demonstrates a simple implementation of how a remote agent desktop client can receive these published GIS data using its own client HTTP server.

Notification Service Example Code

The Notification Service example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
package com.genesyslab.gis.services.notification;
import java.util.Hashtable;

public class NotifyService {
   private static Hashtable messages = new Hashtable();
```

notify() Method

This method's signature follows the requirement set by the GIS `notification.wsdl` file and cannot be changed.

Note: You cannot modify the signature or the notification service will not work.

This method uses a hashtable to store data received from GIS into the cache. This is only a suggestion of how you can store the data, but the code can be changed to use alternative storage method. For example, you could store information in a database or file that the client applications access directly.

```
   public void notify(String eventType, String[][] keyVal) {
```

```
    System.out.println("Hello from gis notification service");
    System.out.println("event type " + eventType);
    for(int i=0; i<keyVal.length;i++)
       for(int j=0;j<keyVal[0].length;j++){
          System.out.println("key val " + keyVal[i][j]);
          if (keyVal[i][j].indexOf("SessionService") != -1) //contains session id
             messages.put(keyVal[i][j],keyVal);
             //store (and overwrite) each session's message using its sid as key value
       }
}
```

retrieveMessage() Method

This method is *not* part of the GIS `notification.wsdl` specification. Its purpose is to demonstrate that you can deploy a user-defined interface in the same notification service to retrieve the data stored by the `notify()` method above. This is just one of many ways to retrieve the data stored.

Note: You do not need to deploy a user service to retrieve stored data. A desktop client can access information through a database or flat file, but web clients that cannot or should not access database or file systems can benefit from this solution.

In the sample `deploy.wsdd` file provided, note that the `allowedMethods` parameter value is set to `*`. This setting allows any method defined in this class to be published as an interface.

The extra API or interface in the service is called `retrieveMessage`. You can see it when you go to this URL:

`http://<your_tomcat_hostName>:<your_tomcat_port>/axis/services`

The page lists all the deployed services with the interface exposed.

```
public String retrieveMessage(String sid){
   try{
      System.out.println("sid= " + sid);
      String[][] msgArray = (String[][])messages.get(sid);
      String msg = new String();
      while (msgArray == null) {
         System.out.println("Waiting to receive information for this session id.");
         msgArray = (String[][])messages.get(sid);
      }
      //flatten the array into a string for ease of demonstration.
      for(int i=0; i<msgArray.length;i++)
         for(int j=0;j<msgArray[0].length;j++) {
            if (j%2 == 1)
               msg = msg + " = " + msgArray[i][j] + "\n";
            else
               msg = msg + msgArray[i][j];
         }
```

```
        return msg;
      }catch(Exception e){return null;}
   }
```

Sample Output

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

```
c:\GCTI\Java Statistic 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:1221036737474H9
     [java] Number of services checked out = 1
     [java] Service checked out = GIS_STATSERVICE
     [java] Statistic needed for 1000
     [java] Statistic of interest is the agent's total login time
     [java] - starting up SimpleAxisServer on port 8000
        (c:\Java Statistic SDK WS Examples)
     [java] Waiting for statistic from GIS server...
     [java]
     [java] Hello from gis notification service
     [java] event type StatisticNotification
     [java] key val timeStamp
     [java] key val 1266154175
     [java] key val sessionId
     [java] key val SOAP:SessionService:1221036737474H9
     [java] key val statId
     [java] key val statref1
     [java] key val date
     [java] key val 1221036865
     [java] key val intervalLength
     [java] key val 1583
     [java] key val lValue
     [java] key val 2
     [java] Hello from gis notification service
     [java] event type StatisticNotification
     [java] key val timeStamp
     [java] key val 1266156159
     [java] key val sessionId
     [java] key val SOAP:SessionService:1221036737474H9
     [java] key val statId
     [java] key val statref1
     [java] key val date
     [java] key val 1221036867
     [java] key val intervalLength
     [java] key val 1585
```

```
[java] key val lValue
[java] key val 4
```

Chapter

8 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. Table 12 lists and describes the methods included in this example.

Table 12: Create Session Methods

Method Name	Method Description
`execute()`	This method contains the main logic of the example. This method logs into the GIS server using the specified host and port information then it retrieves the session from the GIS server.

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;
```

execute() 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 execute(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);
   }
```

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.

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:1221037906825H10
```

Connect Session Service Example

The Connect Session Service example code discussed below is located in the `\Test2\ConnectSesssionServiceExample.cs` file. Table 13 lists and describes the methods included in this example.

Table 13: Connect Session Service Methods

Method Name	Method Description
`execute()`	This method creates a session, subscribes to the Statistics 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;

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

execute() Method

This method uses `CreateSessionExample` to log onto GIS server and retrieve a session. Then it requests a subscription to the GIS Statistics 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:1221037948544H11
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
```

Identify Services Example

The Identify Services example code discussed below is located in the `\Test3\IdentifyServicesExample.cs` file. Table 14 lists and describes the methods included in this example.

Table 14: 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 98.

```
        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:1221038031247H12
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
browsing services available:  GIS_STATSERVICE
releasing services...
Services released:  GIS_STATSERVICE
logout from session
```

Chapter

9 Retrieve Statistical Profile Example—C#

This chapter examines the Retrieve Statistical Profile example.

Statistical profiles are settings that you have defined (using Configuration Manager) in the Stat Server Application object, and contain information such as the time range, time profile, and statistic names available for subscription.

See "Creating Statistic Requests" on page 106 for a short explanation of how to configure a statistic request.

Note: See the *Framework 7.x Stat Server User's Guide* and the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information on statistics configuration.

This chapter includes the following section:

Retrieve Statistical Profile Overview

This example demonstrates how to create session request to retrieve statistical profiles. The response that is returned includes statistic names available for subscription, and any `timerange` and `timeprofile` settings that are available.

Retrieve Statistical Profile Example

The Retrieve Statistical Profile example code is located in the `\test4\RetrieveStatisticalProfileExample.cs` file. Table 15 lists and describes the methods included in this example.

Table 15: Retrieve Statistical Profile Methods

Method Name	Method Description
`execute()`	This method uses the `ConnectSessionServiceExample` class to log in and check out the license for the statistic service. Then it retrieves the statistical profiles to display available statistic names, time ranges, and time profiles.

Retrieve Statistical Profile Example Code

The Retrieve Statistical Profile example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
using System;
using GISServices;

   public class RetrieveStatisticalProfileExample
   {
```

execute() Method

The main logic in the `RetrieveStatisticalProfile` example is contained in the `execute()` method.

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

You must create a new proxy instance.

```
         port_stat = new StatServiceService();
         port_stat.Url = "http://"+targetHost+"/gis/services/
            StatService?GISsessionId="+cs.sid;
```

Send the retrieve statistical profiles request.

```
retrieveStatisticalProfileResponse statistic_profile =
   port_stat.retrieveStatisticalProfile("",
   statisticalProfileType.statisticalProfile);
System.Console.WriteLine("retrieving statistic profiles...");
profileInfo info = statistic_profile.statisticalProfileInfo;
statisticTypeInfoType[] types = info.statisticInfos;
if (types != null)
   foreach( statisticTypeInfoType infoType in types)
   {
      System.Console.Write(infoType.typeName+"( "+infoType.type+") ->");
      foreach( objectType type in infoType.objectTypes)
      {
         System.Console.Write(": "+type);
      }
   }
```

Retrieve filter names.

```
retrieveStatisticalProfileResponse filter_profile =
   port_stat.retrieveStatisticalProfile("",
   statisticalProfileType.filterProfile);
System.Console.WriteLine("retrieving filter profiles...");
profileInfo info1 = filter_profile.statisticalProfileInfo;
parameter[] filters = info1.filters;
if (filters != null)
   foreach( parameter filter in filters)
   {
      System.Console.WriteLine("Key= "+filter.key+" Value= "+filter.value);
   }
```

Retrieve time profile names.

```
retrieveStatisticalProfileResponse time_profile =
   port_stat.retrieveStatisticalProfile("",
   statisticalProfileType.timeProfile);
System.Console.WriteLine("retrieving time profiles...");

profileInfo info2 = time_profile.statisticalProfileInfo;
timeProfile[] timeProfs = info2.timeProfiles;
if (timeProfs != null)
   foreach( timeProfile timeProf in timeProfs)
   {
      System.Console.Write("Key= "+timeProf.key+" Value= "+timeProf.value);
      System.Console.WriteLine(" time interval = "+timeProf.intervalType);
   }
```

Retrieve time range names.

```
retrieveStatisticalProfileResponse time_range_profile =
   port_stat.retrieveStatisticalProfile("",
   statisticalProfileType.timeRangeProfile);
System.Console.WriteLine("retrieving time range profiles...");

profileInfo info3 = time_range_profile.statisticalProfileInfo;
parameter[] timeRanges = info3.timeRanges;
if (timeRanges != null)
   foreach( parameter timeRange in timeRanges)
   {
      System.Console.WriteLine
                     ("Key= "+timeRange.key+" Value= "+timeRange.value);
   }

System.Console.WriteLine("logging out of session...");
```

And finally, send a logout request.

```
cs.port.logout(cs.sid);}
```

Sample Output

The following sample output shows one possible result when running this example:

```
Test 4
logging into session server...
logged into session server successfully.
   Session id =SOAP:SessionService:1221038142341H13
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
retrieving statistic profiles...
ServiceFactor( fValue) ->: Queue: RoutePoint:
   GroupQueuesCurrentPlaceState( unknownValue) ...
GISServices.profileInfo
retrieving time profiles...
Key= OneHour Value= 0:00 1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00
   12:00 13:00
Key= OneMinute Value= 0:00 0:01 0:02 0:03 0:04 0:05 0:06 0:07 0:08 0:09 0:10 0:11
   0:12 0:13 0:
Key= SlidingDay Value= 86400:3600 time interval = SlidingWindow
Key= SlidingHour Value= 3600:600 time interval = SlidingWindow
Key= Default Value= 0:00 time interval = GrowingWindow
retrieving time range profiles ...
Key= Range0-10 Value= 00-10
Key= Range0-120 Value= 0-120
Key= Range0-5 Value= 00-05
Key= Default Value= 0-20
logging out of session...
```

Chapter

10 Retrieve Statistic Example—C#

This chapter presents an example showing how to use the `RetrieveStatistic` method. In particular, it explains what parameters are required to define the statistic you want to retrieve.

This chapter assumes that you are familiar with Stat Server and how statistics are defined. Refer to the *Framework 7.x Stat Server User's Guide* and the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information.

Note: The Statistics SDK Service does not allow access to any statistics that use the `CurrentState` *statistical category.* That is, you cannot collect real-time agent state statistics. However, you can collect statistics that have the `AgentStatus` *subject.*

This chapter includes the following sections:

Retrieve Statistic Overview

The Retrieve Statistic example demonstrates how to do a one-time retrieval of a single statistic. You are required to fully define the desired statistic for each request, as described in "Creating Statistic Requests" on page 106.

Retrieving or Subscribing

If you are retrieving the same statistic many times, you may find it more efficient to subscribe to that statistic instead. When you subscribe to a statistic,

GIS returns a `Statistic` object. You can use this object to retrieve updated statistical values, rather than having to specify all of the statistic parameters manually in each request.

For more information about subscribing to statistics, see Chapter 11 on page 111.

Creating Statistic Requests

To request a statistic, you configure and submit a `Statistic` complex type that is composed of the `statisticID`, `objectIDType`, `metric`, and `schedule` types.

- The `StatisticID` is a unique identifier that GIS uses to track the statistic over time. Your client application is responsible for creating a `statisticID` for a particular statistic.
- The `ObjectIDType` describes the particular object for which the statistic is requested. An `objectID` consists of two strings:
  - `ID`—Specifies one of the configuration objects used by Stat Server. For example, to get statistics about an Agent Group, use the name of the Agent Group; to get statistics about a DN, use the format *<dn name>@<switch name>*.
  - `tenantName`—Specifies the tenant name, as configured in Configuration Manager.
- A `Metric` specifies parameters for filtering Stat Server data. A `Metric` comprises at least an interval (`timeInterval`) and either a `typeName` string or a `statisticType`.

  The `timeInterval` uses a string to define a window (growing, sliding, sliding selection), and may also specify a `length` integer, `slidelength` integer, or `timeProfileName` depending on the type of window. If a `typeName` is given, then that string must match a name known to Stat Server. If a `statisticType` is included, then the application must supply the data needed for the `statisticType` elements, which resolve to strings that must match their counterparts known to Stat Server.

  A `Metric` may also optionally specify the name of a filter or of a comma-separated `TimeRange` list, as well as the data for a `timeRange` type (two integers: one for left time, the other for right time).
- A `Schedule` directs Stat Server to provide a statistic at a certain interval over time. A `Schedule` has a `timeout` and an `insensitivity` value (both integers), as well as a `notificationMode` that may be `ChangesBased`, `TimeBased`, or `ResetBased`.

Note: Refer to the *Framework 7.x Stat Server User's Guide* for more information about the data required to identify any particular statistic.

Retrieve Statistic Example

The Retrieve Statistic example code is located in the `\Test5\RetrieveStatisticExample.cs` file. Table 16 lists and describes the methods included in this example.

Table 16: Retrieve Statistic Methods

Method Name	Method Description
`execute()`	This method creates a session and logs in, and then creates and submits a statistic request for an agent's total login time.

Retrieve Statistic Example Code

The Retrieve Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
using System;
using System.Threading;
using GISServices;

   public class RetrieveStatisticExample
   {
      public StatServiceService port_stat;
      public RetrieveStatisticExample(){}
```

execute() Method

This method uses the `ConnectSessionServiceExample` to log in and retrieve the Session ID, and then creates a statistic request. The method defines specific information about this statistic most of which are quantitative.

See the *Framework 7.x Stat Server User's Guide* or the *Reporting Technical Reference Guide for the Genesys 7.2 Release* for more information on statistics configuration.

Note: This example calls the `retrieveStatistic()` method twice. GIS will initialize the statistic when this method is called for the first time. After a 32-second timeout, a second call to this method will retrieve the true value.

```
public void execute(String targetHost)
{
```

This example uses the `ConnectSessionServiceExample` to log in to GIS, register for a service, and retrieve the Session ID.

```
ConnectSessionServiceExample cs = new ConnectSessionServiceExample();
cs.execute(targetHost);
```

Now create a new proxy instance.

```
port_stat = new StatServiceService();
port_stat.Url = "http://"+targetHost+"/gis/services/
   StatService?GISsessionId="+cs.sid;
```

Next build a statistic object.

```
statistic _statistic = new statistic();
_statistic.statisticId = "statref1";
```

To define the statistic, start by building the Object ID.

```
objectIdType _objectIdType = new objectIdType();
System.Console.WriteLine("Statistic needed for Agent 1000");
```

Then build the statistic by entering the Employee ID for the agent you are interested in, the Tenant name, and the Object ID data.

```
_objectIdType.id = "1000";
_objectIdType.tenantName = "defaultTenant";
_statistic.objectId = _objectIdType;
```

Define the metrics, such as time interval, and add the metric specifications to the statistic definition.

```
metric _metric = new metric();
statisticType _statisticType = new statisticType();
objectType[] ot = new objectType[]{objectType.Agent};
_statisticType.objectType = ot;
_metric.statisticType = _statisticType;
System.Console.WriteLine("Statistic interested is agent's total login time");
_metric.typeName = "TotalLoginTime";
timeInterval _timeInterval = new timeInterval();
_timeInterval.intervalType = timeIntervalType.GrowingWindow;
_metric.interval = _timeInterval;
_statistic.metric = _metric;
```

Specify the statistic schedule: time based, reset based, or changes based.

```
        schedule _schedule = new schedule();
        _schedule.notificationMode = scheduleMode.ChangesBased;
```

Add the schedule data to the statistic definition.

```
        _statistic.schedule = _schedule;
```

Now retrieve the statistic.

```
        retrieveStatisticResponse response = port_stat.retrieveStatistic(_statistic,
   "StatServer");
```

Note: The first call to `retrieveStatistic` simply initializes the statistic, and will always return a value of zero (0). You must add a timeout and then retrieve the statistic value a second time to get the actual value.

The GIS default session timeout is 3.6 seconds, but you can specify other values such as 50 seconds as shown in this example.

```
        Thread.Sleep(31000);
        response = port_stat.retrieveStatistic(_statistic, "StatServer");
```

Now GIS returns the statistic value it has retrieved from Stat Server.

```
        statisticValue val = (statisticValue)response.statisticValue;
System.Console.WriteLine("Retrieved statistic value "+((eventValue)val.eventValues.
                                                              GetValue(0)).LValue);
   }
```

Sample Output

The following sample output shows one possible result when running this example:

```
Test 5
logging into session server...
logged into session server successfully.
Session id =SOAP:SessionService:1221038265966H14
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
Statistic needed for Agent 1000
Statistic interested is agent's Total Login Time
Retrieved statistic value = 1433
```

Chapter

11 Subscribe and Retrieve Examples—C#

This chapter examines the subscribe to/retrieve statistics examples. This chapter includes the following sections:

Subscribing to Statistics

Rather than creating and submitting multiple individual requests to track changes in the value of a statistic over time, you can subscribe to an available statistic. This provides a `Statistic` object that allows you to check for statistical updates quicker and easier than if you had to specify all of the statistic parameters for each request.

Creating a subscription

To create a subscription to a statistic, you specify all of the parameters required to build a subscription object and then use the `subscribeStatistic()` method to send a subscription request to the GIS statistics session.

Once the subscription has been created, you can reference it using a `subscription` object with the following properties:

- `statisticID`—The unique identifier for a statistic, assigned by you when created the subscription.

- `scope`—Corresponds to the number of values for each particular `statisticID`, calculated between one time the application subscribes and the next. The application can retrieve a set of last values for both subscription instances and use them as the scope.

To remove a subscription, use the unique `statisticID` with the `subunsubscribeStatistic` request.

Retrieving the Value of a Subscribed Statistic

To retrieve the value of a subscribed statistic, you send a list of statistic subscriptions and a `Notification` object to the `retrieveSubscribedStatistics` request.

The `retrieveSubscribedStatistics` request is limited by a restriction time limit. Your client application must wait for the period (in seconds) set by the `restriction_time` option in the GIS Application object `Options` tab before sending another retrieve request. You can adjust the length of this setting from Configuration Manager, as described in the *Genesys Integration Server 7.6 Deployment Guide*.

Subscription Notification Types

If your statistic subscription request `unsolicitedNotification` type includes the URL for your client application's web server as its value, GIS sends unsolicited notification whenever the data for your subscribed statistics is updated. For details about unsolicited notification, see Chapter 12 on page 119.

If the `unsolicitedNotification` type is null, then you must set the notification mode to either:

- `Polling`
- `Blocked`

For details about notification modes, see Chapter 1, "Notification Modes" on page 21.

Overview of Examples

This chapter discusses the following examples:

- Subscribe To Statistic Example—This example demonstrates how to subscribe and unsubscribe to a statistic. Most of the code for this example is required to build a `statistic` object used in the subscribe request.
- Retrieve Subscribed Statistic Example—This example demonstrates how to retrieve a subscribed statistic using the Polling mode of solicited notification. The example uses the Subscribe To Statistic example to subscribe and unsubscribe to the statistic.

  In this example, the object of interest is an agent and the statistic retrieved is the total login time for that agent. We are also assuming the agent object is under the Tenant Resources. This example can be modified slightly to retrieve statistics for Agents, Queues, Places, or other object types.

Subscribe To Statistic Example

The Subscribe To Statistic example code is located in the `\Test6\SubscribeToStatisticExample.cs` file. Table 17 lists and describes the methods included in this example.

Table 17: Subscribe To Statistic Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to a statistic.
`unsubscribe()`	This method is used to unsubscribe from the statistic created in the `subscribe()` method.

Subscribe To Statistic Example Code

The Subscribe To Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
using System;
using GISServices;

   public class SubscribeToStatisticExample
   {
      public StatServiceService port_stat;
```

execute() Method

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

This method uses the `ConnectSessionServiceExample` to log in and retrieve the Session ID, and then subscribes to a statistic. The statistic subscribed in this example is for an agent's total login time.

```
      public void execute(String targetHost)
      {
```

First, it uses the `ConnectSessionServiceExample` to log in to GIS, register for a service, and retrieve the Session ID.

```
         ConnectSessionServiceExample cs = new ConnectSessionServiceExample();
         cs.execute(targetHost);
```

Now create a new proxy instance.

```
port_stat = new StatServiceService();
port_stat.Url = "http://"+targetHost+"/gis/services/
   StatService?GISsessionId="+cs.sid;
```

Next build a statistic object.

```
statistic _statistic = new statistic();
_statistic.statisticId = "statref1";
```

To define the statistic, start by building the Object ID.

```
objectIdType _objectIdType = new objectIdType();
System.Console.WriteLine("Statistic needed for Agent 1000");
```

Then build the statistic by entering the Employee ID for the agent you are interested in, the Tenant name, and the Object ID data.

```
_objectIdType.id = "1000";
_objectIdType.tenantName = "defaultTenant";
_statistic.objectId = _objectIdType;
```

Define the metrics, such as time interval, and add the metric specifications to the statistic definition.

```
metric _metric = new metric();

statisticType _statisticType = new statisticType();
objectType[] ot = new objectType[]{objectType.Agent};
_statisticType.objectType = ot;
_metric.statisticType = _statisticType;
System.Console.WriteLine("Statistic interested is agent's total login time");
_metric.typeName = "TotalLoginTime";
timeInterval _timeInterval = new timeInterval();
_timeInterval.intervalType = timeIntervalType.GrowingWindow;
_metric.interval = _timeInterval;
```

Add the metric data to the statistic.

```
_statistic.metric = _metric;
```

Specify the statistic schedule: time based, reset based, or changes based.

```
schedule _schedule = new schedule();
_schedule.notificationMode = scheduleMode.ChangesBased;
```

Add the schedule data to the statistic definition.

```
   _statistic.schedule = _schedule;
```

Next, subscribe to the statistic.

```
   port_stat.subscribeStatistic(_statistic, "StatServer", new
      unsolicitedNotification());
   System.Console.WriteLine("Successfully subscribed to statistic.");
}
```

unsubscribe() Method

This method is used to unsubscribe from the `statref1` statistic created in the `execute()` method.

```
public void unsubscribe()
{
   port_stat.unsubscribeStatistic("statref1");
   System.Console.WriteLine("Unsubscribed to statistic.");
}
```

Sample Output

The following sample output shows one possible result when running this example:

```
Test 6
logging into session server...
logged into session server successfully.
Session id =SOAP:SessionService:1221038339559H15
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
Statistic needed for Agent 1000
Statistic interested is agent's Total Login Time
Successfully subscribed to statistic.
Unsubscribed to statistic.
```

Retrieve Subscribed Statistic Example

The Retrieve Subscribed Statistic example code is located in the `\Test7\RetrieveSubscribedStatisticExample.cs` file. Table 18 on page 116 lists and describes the methods included in this example.

Table 18: Retrieve Subscribed Statistic Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then retrieves the subscribed statistic using the Polling notification mode.

Retrieve Subscribed Statistic Example Code

The Retrieve Subscribed Statistic example code is listed below, with comments and analysis added to focus your attention on important details or clarify unusual processes.

```
using System;
using GISServices;

   public class RetrieveSubscribedStatisticExample
   {
```

execute() Method

This method retrieves the subscribed statistic using the `polling` notification mode. The code uses the `SubscribeToStatisticExample` to establish a session and subscription to the statistic `Total_Login_Time`, as specified in `SubscribeToStatisticExample`.

```
      public void execute(String targetHost)
      {
         SubscribeToStatisticExample sb = new SubscribeToStatisticExample();
         sb.subscribe(targetHost);
```

This retrieves the subscribed statistic:

```
         statisticSubscriptions subscriptions = new statisticSubscriptions();
         statisticSubscription subscription = new statisticSubscription();
         subscription.scope = "all";
         subscription.statisticId = "statref1";
         statisticSubscription[] array_subscriptions = new statisticSubscription[1];
         array_subscriptions.SetValue(subscription,0);
         subscriptions.statisticSubscription = array_subscriptions;
```

You must next specify the notification mode. The example uses the `polling` mode.

```
        notification notif = new notification();
        notif.mode = notificationMode.Polling;
        notif.timeout = "10";
        retrieveSubscribedStatisticsResponse response =
           sb.port_stat.retrieveSubscribedStatistics(subscriptions, notif);
```

Next, the example returns the value.

```
        statisticValue val = (statisticValue)response.statisticValues.GetValue(0);
        System.Console.WriteLine("Retrieved statistic value = "+
           ((eventValue)val.eventValues.GetValue(0)).LValue);
    }
  }
```

Sample Output

The following sample output shows one possible result when running this example:

```
Test 7
logging into session server...
logged into session server successfully.
Session id =SOAP:SessionService:1221038371621H16
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
Statistic needed for Agent 1000
Statistic interested is agent's Total Login Time
Successfully subscribed to statistic.
Retrieved statistic value = 1507
```

Chapter

12 Unsolicited Notification Example—C#

This chapter examines the Unsolicited Notification example.This chapter includes the following sections:

About Unsolicited Notification

GIS provides an unsolicited notification service feature for the Statistics SDK Service. This feature allows GIS to transmit updates to subscribed statistics to a designated client-side HTTP server that uses a notification service to distribute the updated information to client applications.

The examples included in this release provide an example of one way you might implement the Unsolicited Notification feature. The mandatory requirements for using unsolicited notification are:

- An HTTP-capable server to receive notification updates.
- Implementation of the notification WSDL on the HTTP server.

Note: Genesys recommends that you be familiar with developing web services before attempting to implement unsolicited notification.

Unsolicited Event Structure

Once you subscribe to the statistic, the client starts to receive notification events from GIS. A sample notification event (HTTP Request) that could be sent by GIS is included below for your reference:

```
POST /notifSOAPSOAP HTTP/1.1 Content-Type: text/xml; charset="utf-8" SOAPAction: notify
   User-Agent: Java1.3.1_01 Host: <client_host>:<remoting_port>
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2 Connection: keep-alive
   Content-length: 1367
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:tns="http://www.genesyslab.com/notification"
     xmlns:types="http://www.genesyslab.com/notification/encodedTypes"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <tns:notify>
      <eventType xsi:type="xsd:string">StatisticNotification</eventType>
      <keyVal href="#id1" />
    </tns:notify>
    <soapenc:Array id="id1" soapenc:arrayType="xsd:string[][5]">
      <Item href="#id2" />
      <Item href="#id3" />
      <Item href="#id4" />
      <Item href="#id5" />
      <Item href="#id6" />
    </soapenc:Array>
    <soapenc:Array id="id2" soapenc:arrayType="xsd:string[2]">
      <Item>sessionId</Item>
      <Item>SessionService:1040391220078H5</Item>
    </soapenc:Array>
    <soapenc:Array id="id3" soapenc:arrayType="xsd:string[2]">
      <Item>statId</Item>
      <Item>statref1</Item>
    </soapenc:Array>
    <soapenc:Array id="id4" soapenc:arrayType="xsd:string[2]">
      <Item>date</Item>
      <Item>1040391882</Item>
    </soapenc:Array>
    <soapenc:Array id="id5" soapenc:arrayType="xsd:string[2]">
      <Item>intervalLength</Item><Item>386</Item>
    </soapenc:Array>
    <soapenc:Array id="id6" soapenc:arrayType="xsd:string[2]">
      <Item>lValue</Item>
      <Item>366</Item>
    </soapenc:Array>
  </soap:Body>
</soap:Envelope>
```

This example request is composed of a `StatisticNotification` notification type and a `Keyval` array that contains the following values:

- `sessionId`—Client `SessionId` to which the current statistic belongs.

> **Note:** The client-side code should filter the notification messages based on the current sessionId to filter out previously-cached events.

- `statId`—The unique `statisticID` value for the current statistic.
- `date`—Date when the statistic value was issued from Stat Server.
- `intervalLength`—Length of the interval that is received from Stat Server.
- `lvalue`, `fvalue`—The statistic value, as calculated by Stat Server.

Note: Each time a notification is successfully sent to a client, that client's `sessionTimeout` timer is rescheduled. Once a notification attempt fails (because the value specified by the `maxAttempts` parameter is exceeded), the `sessionTimeout` is no longer rescheduled.

You can set the `maxAttempts` value by editing your `modules.conf` configuration file. For additional details about this setting, see the *Genesys Integration Server 7.6 Deployment Guide*.

Subscribe To Unsolicited Notification Example

The Unsolicited Notification example code is located in the `\Test8\UnsolicitedNotificationExample.cs.cs` file. Table 19 lists and describes the methods included in this example.

Table 19: Unsolicited Notification Methods

Method Name	Method Description
`execute()`	This method creates a session, logs into a GIS server, and subscribes to the licensed services. It then requests an unsolicited notification server from GIS.
onNewNotif()	This listener method is called from the `InteropNS.cs` file when a new statistic notification is received.
unsubscribe()	Unsubscribes from the statistic and releases the reference.

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 unusual processes.

```
using System;
using GISServices;
   public class UnsolicitedNotificationExample
   {
      public StatServiceService port_stat;
      public String remotingPort = "8000";
```

execute() Method

This method uses the `ConnectSessionServiceExample` to log in and retrieve the Session ID. Then it subscribes to a statistic, `TotalLoginTime`.

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

Create a new proxy instance.

```
         port_stat = new StatServiceService();
         port_stat.Url = "http://"+targetHost+"/gis/services/
            StatService?GISsessionId="+cs.sid;
```

Build a statistic object.

```
         statistic _statistic = new statistic();
         _statistic.statisticId = "statref1";
```

Build the Object ID, including the Employee ID for the agent in which you are interested and the Tenant under which the Agent object is located.

```
         objectIdType _objectIdType = new objectIdType();
         System.Console.WriteLine("Statistic needed for Agent 1000");
         _objectIdType.id = "1000";
         _objectIdType.tenantName = "defaultTenant";
```

Now add the Object ID data to the statistic.

```
         _statistic.objectId = _objectIdType;
```

Build metric data.

```
metric _metric = new metric();

statisticType _statisticType = new statisticType();
objectType[] ot = new objectType[]{objectType.Agent};
_statisticType.objectType = ot;
_metric.statisticType = _statisticType;
System.Console.WriteLine("Statistic interested is agent's total login time");
_metric.typeName = "TotalLoginTime";
timeInterval _timeInterval = new timeInterval();
_timeInterval.intervalType = timeIntervalType.GrowingWindow;
_metric.interval = _timeInterval;
```

Then add metric data to the statistic.

```
_statistic.metric = _metric;
```

Now specify the statistic update schedule.

```
schedule _schedule = new schedule();
_schedule.notificationMode = scheduleMode.ChangesBased;
```

And then add the schedule data to the statistic.

```
_statistic.schedule = _schedule;

unsolicitedNotification un = new unsolicitedNotification();
String remotingHost = System.Net.Dns.GetHostName();
un.url = "http://"+remotingHost+":"+remotingPort+"/notifSOAPSOAP";
```

At this point, create a new `HttpServerChannel` to receive messages from GIS.

```
System.Runtime.Remoting.Channels.Http.HttpServerChannel channel = new
   System.Runtime.Remoting.Channels.Http.HttpServerChannel("channel" +
   remotingPort, Int32.Parse(remotingPort));
```

Now register the channel.

```
System.Runtime.Remoting.Channels.ChannelServices.RegisterChannel(channel);
System.Runtime.Remoting.RemotingConfiguration.RegisterWellKnownServiceType(
   System.Type.GetType("InteropNS.notifSOAPSOAP"), "notifSOAPSOAP",
      System.Runtime.Remoting.WellKnownObjectMode.Singleton);
System.Runtime.Remoting.Channels.IChannel notifchannel =
   (System.Runtime.Remoting.Channels.IChannel)System.Runtime.
   Remoting.Channels.ChannelServices.RegisteredChannels.GetValue(0);
```

Next subscribe to the statistic.

```
port_stat.subscribeStatistic(_statistic, "StatServer", un);

System.Console.WriteLine("Successfully subscribed to statistic.");
System.Console.WriteLine("Press a key to stop receiving events");
System.Console.ReadLine();
```

onNewNotif() Method

This listener method is called from the `\Test8\InteropNS.cs` file when a new statistic notification is received.

```
public static void onNewNotif(String eventType, String[][] keyVal)
{
   System.Console.WriteLine("New notification ("+eventType+") received");
   foreach( String[] key in keyVal)
   {
      foreach( String value in key)
      {
         System.Console.Write(value+" ");
      }
      System.Console.WriteLine();
   }
}
```

unsubscribe() Method

Here we unsubscribe to a statistic and release the reference.

```
public void unsubscribe()
{
   //Unsubscribe statistic
   port_stat.unsubscribeStatistic("statref1");
   System.Console.WriteLine("Unsubscribed to statistic.");
}
```

The InteropNS File

This is a file generated by the .NET Framework tool. It is used when registering the channel through which your client application receives unsolicited notification updates from GIS.

```
using System;
using System.Runtime.Remoting.Messaging;
using System.Runtime.Remoting.Metadata;
```

```
using System.Runtime.Remoting.Metadata.W3cXsd2001;
/*
 * File generated by .Net Framework tool:
 * soapasuds -url:http://<gis_host>:<gis_port>/gis/services/CSProxyService?wsdl -gc
 */

namespace InteropNS {
   [Serializable, SoapType(XmlElementName=@"notifSOAPSOAP",
      XmlNamespace=@"http://www.genesyslab.com/notification.wsdl",
      XmlTypeName=@"notifSOAPSOAP",
      XmlTypeNamespace=@"http://www.genesyslab.com/notification.wsdl")]
   public class notifSOAPSOAP : System.MarshalByRefObject
   {
      [SoapMethod(SoapAction=@"notify",ResponseXmlElementName=@"notify",
         XmlNamespace=@"http://www.genesyslab.com/notification.wsdl",
         ResponseXmlNamespace=@"http://www.genesyslab.com/notification.wsdl")]
```

notify() Method

This is the call to the notification listener class.

```
      public void notify(String eventType, String[][] keyVal)
      {
         UnsolicitedNotificationExample.onNewNotif(eventType, keyVal);
         return;
      }...
```

Sample Output

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

```
Test 8
logging into session server...
logged into session server successfully.
Session id =SOAP:SessionService:1221038561214H17
Number of services subscribed = 1
Service subscribed = GIS_STATSERVICE
Statistic needed for Agent 1000
Statistic interested is agent's Total Login Time
Successfully subscribed to statistic.
Press a key to stop receiving events
New notification (StatisticNotification) received
timeStamp 1267850728
sessionId SOAP:SessionService:1221038561214H17
statId statref1
date 1221038561
intervalLength 3279
```

```
lValue 1698
New notification (StatisticNotification) received
timeStamp 1267852712
sessionId SOAP:SessionService:1221038561214H17
statId statref1
date 1221038563
intervalLength 3281
lValue 1700
```

Index

J

L

M

N

P

R

S

T

U

V

W

X