GENESYS
AN ALCATEL-LUCENT COMPANY

Workforce Management 7.6

WFM Integration API

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)-118-974-7002	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: 76wm_dev_02-2008_v7.6.001.00

Table of Contents

Preface

Welcome to the *Workforce Management 7.6 WFM Integration API Developer's Guide*. This guide provides a high-level overview of Workforce Management (WFM) 7.5 WFM Integration API (application programming interface) features and functions, together with software-architecture information and development-planning materials. It presents an overview of the API's architecture and communication protocols, the setup procedures for client development, and the product's API capabilities.

This document is valid only for the 7.5 release of this product.

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

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

The WFM Integration API is built around the `wfmapi.java` library, which presents an API for developing third-party media applications. The library provides connectivity with WFM Server, so that your applications can view, request, and update WFM objects and data.

Intended Audience

This guide is primarily intended for developers who are familiar with Simple Object Access Protocol (SOAP), Hypertext Transfer Protocol (HTTP), and

XML (Extensible Markup Language) technologies. It assumes that you have a basic understanding of:

- Network design and operation.
- Your own network configurations.

Depending on the technology that you choose for client development, you require a working knowledge of Java or of some other Web Services client-side programming language.

You should also be familiar with the Genesys Framework and with WFM 7.5 features.

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 opening chapter, this guide contains these chapters:

- Chapter 1, “About the WFM Integration API,” on page 15, provides an overview of the API, its architecture, and the functionality it can provide. It also lists the supported platforms and development tools.

- Chapter 2, "Locator, Session, and Util," on page 23, explains how to use the Locator service to identify the WFM Server with which to open a session; how to open, manage, and close sessions; and how to use the `OleDateTime` class of the Util service, which is used in requests to all the other services.
- Chapter 3, "Configuration," on page 29, introduces the Configuration service functions.
- Chapter 4, "Calendar," on page 33, introduces the Calendar service functions.
- Chapter 5, "Forecast," on page 37, introduces the Forecast service functions.
- Chapter 6, "Schedule," on page 41, introduces the Schedule service functions.
- Chapter 7, "Schedule Trading," on page 51, introduces the Schedule Trading service functions.
- Chapter 8, "Performance," on page 69, introduces the Performance service functions.
- Chapter 9, "Adherence," on page 73, introduces the Adherence service functions.

Document Conventions

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

Document Version Number

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

75wm_dev_04-2006_v7.5.000.00

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:

- *Workforce Management 7.5 WFM Integration API Reference* (JavaDoc), which contains a complete description of the API, its services, classes, and methods.
- *Workforce Management 7.5 Administrator’s Guide*, which provides an overview of the Genesys Workforce Management product, including its architecture; presents deployment-planning considerations; explains how to install and configure, start, and stop all WFM components; explains how to use the WFM Database Utility; provides troubleshooting suggestions; and includes a list of terms used in Genesys WFM, with definitions.
- *Workforce Management 7.5 Configuration Utility Help*, which explains how to use the WFM Configuration Utility to set up Workforce Management objects, such as sites, time zones, and activities; and constraints, such as security settings, working hours, and time-off accrual rules.
- *Workforce Management 7.5 Web for Supervisors Help*, which explains how to use the Supervisors web interface to create forecasts and schedules, make schedule and staffing changes, and to track agent real-time adherence and contact center performance. WFM Web for Supervisors also contains the Workforce Management reporting functions.
- *Workforce Management 7.5 Web for Agents Help,* which instructs agents on how to check their schedules; request time off, working hours, and other preferences; and make schedule trades with other agents.
- The *Genesys Technical Publications Glossary*, which ships on the Genesys Documentation Library CD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.

- The *Genesys 7 Migration Guide*, also on the Genesys Documentation Library CD, 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 CD, 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 WFM Integration API

This chapter introduces the Workforce Management (WFM) Integration API (application programming interface), its components, features, and scope of use.

This chapter includes the following sections:

Features Overview

The WFM Integration API enables you to build .NET or Java applications that can:

- View certain WFM objects, data, and agent status.
- Retrieve certain data for use in third-party applications.
- Update WFM with certain data drawn from third-party applications.

Note: Not all WFM functions and features are available through the WFM Integration API.

Your client-side applications use SOAP to communicate with the API. To develop successful client applications for WFM Integration API, you can:

- Use the Microsoft .NET Framework SDK, version 1.1, to create stubs for a C#-based application.
- Use the Apache AXIS toolkit, version 1.2.1, to create stubs for a Java-based application.

Components

The documentation for the WFM Integration API product includes the following components, both of which are delivered on the Documentation Library CD.

- The *Workforce Management 7.5 WFM Integration API Reference* in JavaDoc format.
- This *Developer's Guide*.

The namespace for the WFM Integration API is:

- `com.genesyslab.wfm7.API`

The WFM Integration API includes the following packages. Where possible, use the package version that matches the release of WFM you have deployed.

- `service.adherence7xx`—Exposes the interface that enables you to retrieve historical agent-adherence data.
- `service.builder720`—Builds schedule for the specified schedule scenario site.
- `service.calendar7xx`—Exposes the interface that enables you to retrieve and, in some cases, update data about Calendar items, such as exceptions and time off.
- `service.config7xx`—Exposes the interface that enables you to retrieve and, in some cases, update data about WFM configuration objects, such as agents, activities, sites, and users.
- `service.forecast7xx`—Exposes the interface that enables you to retrieve forecast data.
- `service.locator`—Locates the WFM Server with which you will establish your session. Can also check login IDs. You must always use this service first when you are opening a session.
- `service.performance7xx`—Exposes the interface that enables you to retrieve contact-center performance data.
- `service.schedule7xx`—Exposes the interface that enables you to retrieve and, in some cases, update WFM agent-schedule data, such as agent daily schedules and agent scheduled states.
- `service.scheduleTrade7xx`—Exposes the interface that enables you to retrieve data about schedule trade proposals, and to accept or decline them.

- `service.session700`—Creates a session with the WFM Server identified by the Locator service. Also manages and closes sessions. You must always use the session service to establish your session before using any of the other services and to close your session when you are done.
- `util`—Exposes the interface that sets the date and time used in all other methods.

The Discovery Interface

The Discovery Interface lists all of the services included in the WFM Integration API.

Note: This interface is particularly useful for .NET developers who have to create C# stubs. Java stubs are included in the `wfmapi.jar` file.

To view the interface, enter the following URL into your browser's address bar:

```
http://<WFM Server host name>:<WFM Server port number>/?Handler=DISCO
```

Warning! The Discovery Interface page lists all of the services included in the API. This list includes outdated services. Developers should use only the most recent version of each service (the one with the highest version number).

Platform Requirements

The platform requirements for developing your application are a little different from the platform requirements for your final application in production.

Development Platform

For .NET development, you need:

- Microsoft .NET Framework SDK, version 1.1 or higher (available at `http://msdn.microsoft.com/netframework/`).
- Microsoft Visual Studio .NET 2003.

For Java development, you need:

- Apache AXIS toolkit, version 1.2.1 (available at `http://xml.apache.org/axis/index.html`).
- Java Development Kit (JDK), version 1.3 or higher.

Production Runtime Platform

For .NET development, you need:

- Microsoft .NET Framework version 1.1 or higher (available at `http://msdn.microsoft.com/netframework/`).

For Java development, you need:

- Apache AXIS toolkit, version 1.1 (available at `http://xml.apache.org/axis/index.html`).
- Java Runtime Environment (JRE), version 1.3 or higher.

Architecture

The WFM Integration API provides you with a service-oriented API. Through the services exposed in this API, your application can establish a session with WFM Server. You then send your requests to that WFM Server, which retrieves the data from the WFM Database. Figure 1 on page 18 shows this architecture.

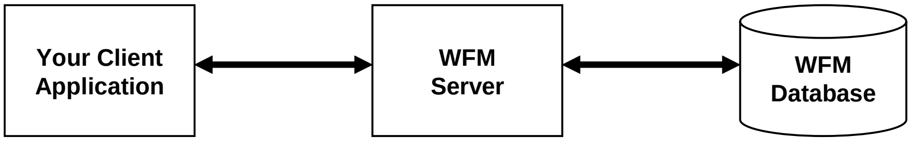

Figure 1: WFM Integration API Architectural Overview

On the server side, the exposed services perform the client’s requests.

If you have an environment with multiple WFM Servers, the WFM Server instance to which you send your initial Locator service request identifies the least-busy WFM Server and returns its URL so that you can establish a session with that WFM Server. It is not necessarily the same one you originally sent your Locator service request to.

Service-Oriented Architecture

The Service-Oriented Architecture (SOA) is a specific type of distributed system in which features are exposed through services. When you are using the WFM Integration API, you are dealing with service interfaces that do not manage anything locally. Each service defines a specific feature of your distributed system.

The message protocol is based on SOAP, which, in turn, is based on HTTP.

Asynchronous Requests

Some processes may take significant time. WFM Server has separate methods for initiating a long process, canceling the process, retrieving process status, retrieving process progress, and retrieving results.

Asynchronous methods take the user parameters, start the process, and return an immediate response. Later, you can request WFM Server to send the status (progress) of the process. To retrieve the status and progress of the request, use the Session service.

Note: Check the progress of an asynchronous request by calling `GetRequestProgress.` Call `GetRequestResult` only after the returned progress is 100%.

Synchronous methods start the process and return the result after the process has been completed.

Using the Code Samples

The WFM Integration API includes a series of code examples in Java. It also includes a short example in .NET that contains no actual functions but orients C# developers in how to use the API.

When you install WFM API, the following files are placed into these directories:

jar file(s)	`\<Genesys product dir>\<WFMnn dir>\api\lib`
javadoc files	`\<Genesys product dir>\<WFMnn dir>\api\docs`
Java samples	`\<Genesys product dir>\<WFMnn dir>\api\samples\java`
.NET samples	`\<Genesys product dir>\<WFMnn dir>\api\samples\NET`

...where `<WFMnn dir>` is the top directory of all installations of all Workforce Management components on Windows by default and *nn* is the current release number, such as 72 or 75 or 76.

You can view the code samples using an ASCII viewer such as Notepad.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Viewing the Code Samples

The samples are intended mainly to be used as starting points for building your own application. There is at least one code sample for each service.

Note: The `SessionConnect.java` sample demonstrates use of both the Locator service and the Session service.

To use the samples as starting points:

1. Open a sample file.
2. Review the methods used and select those you are interested in.
3. Replace variables with those specific to your environment. The variables include:
   - The host name and port number for your WFM Server.
   - The `Application` object name of your client application as configured in Configuration Manager.
   - The object IDs used in your WFM database (such as site IDs, agent IDs, and so on).
   - Dates and times.

Note: Refer to the *Workforce Management 7.5 WFM Integration API Reference*, a JavaDoc file located on the Genesys Technical Support website and the Documentation Library CD, for a complete description of all publicly-exposed methods in the WFM Integration API.

Running the Code Samples

To run the code samples you must:

1. Install the Apache Axis 1.2.1 toolkit.
2. Put the `wfmapi.jar` and the Axis `.jar` files in your classpath. See the Apache Axis documentation for help identifying the Axis `.jar` files you should put in the classpath.
3. Modify all environment-specific variables to fit your environment. The variables include:
   - The host name and port number for your WFM Server.
   - The `Application` object name of your client application as configured in Configuration Manager.
   - The object IDs used in your WFM database (such as site IDs, agent IDs, and so on).
   - Dates and times.

Using the WFM Integration API Reference

The WFM Integration API Reference is delivered in JavaDoc format. It is available on the Genesys Technical Support website and on the Documentation Library CD.

To use the API Reference:

1. Locate the zip file containing the JavaDoc.
2. Extract the zip file to a local or network directory.
3. Double-click the `index.html` file to open the JavaDoc.

The API Reference contains descriptions of each service (also known as a package).

- Follow the links to locate whatever information you require.

Note: The JavaDoc includes some methods that are for Genesys internal use only. These methods have no comments attached to them.

Chapter

2 Locator, Session, and Util

This chapter explains the Locator, Session and Util services, represented by the `WFMLocatorServiceSoap` and `WFMSessionService700Soap` interfaces.

- The Locator service identifies the WFM Server with which you should establish your session. It also can check login IDs.
- The Session service opens, manages, and closes sessions. It also enables you to track requests, close completed requests, and cancel no-longer-needed requests.
- The Util service has one class, `OleDateTime`, that is used in all the other services whenever you are using a date or a date/time parameter.

This chapter includes the following sections:

Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15.

Locator Essentials

The Locator service is presented through the WFMLocatorServiceSoap interface. Use this service to find the WFM Server with which you will establish a session, and to return its URL for you to use in subsequent requests.

Use the Locator service once per session, just before you establish your session. Once the session is established, your client application always works with the same WFM Server.

If your WFM deployment uses more than one WFM Server, the Locator service identifies the least busy WFM Server and establishes your session with that server, thus performing a basic form of load-balancing. The load balancing is done not per request, but per session.

The Locator service can also check the user login ID and password, to verify whether the user is a supervisor or an agent, and whether his or her login information matches the information in the WFM Database.

method LocateServer

The method locates the best WFM server with which to establish your session. The parameters you specify are taken to account to determine the best server. The WFM Server that serves as locator may return its own URL as the value for serverLocation or that of a different WFM Server.

Once you have established a session with the WFM Server indicated by the serverLocation, your client should not call locateServer again, but should go directly to WFM Server, which maintains the session.

method locateServerLogin

Use locateServerLogin, if your client application already knows the username and password of the user. When you use this method, you do not need to fetch the DBID of the user’s Person object before calling this method. It checks whether the username and password specified are correct, provides the Person DBID, and uses the isAgent flag to indicate whether the user is an agent or a supervisor. And, like locateServer, locateServerLogin directs your client application to WFM Server so you can establish your session.

The SessionConnect.java code sample shows how to use the Locator service while establishing a session. See “Using the Code Samples” on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Session Essentials

The Session service is presented through the `WFMSessionService700Soap` interface. Use this service to open and log into a session, monitor and manage request results, and close your session.

When you open your session, you receive a session ID that must be included in all subsequent requests during that session.

Note: The Locator service does not require a session ID.

method openSession

To use `openSession`, your client application must send the DBID of its own `Application` object as configured in Configuration Manager to open the session. Only supervisors who are recognized by the WFM User Security subsystem can open a session.

method PingSession

Sends a keep alive signal for the specified session if the WFM Server is about to close the session because of lack of activity from your client application for a specified timeout period. This method has no parameters.

method CloseSession

Closes your open session. This method also closes all open snapshots that are associated with this session. If other requests belonging to your session are being executed in parallel to `closeSession`, then the session is closed when the last request has been served. However, new requests for this session are not served after WFM Server receives the `closeSession` call. This method has no parameters.

Snapshots

The Session service provides a common method for closing already-opened snapshots. Snapshots are opened by other services.

Asynchronous Requests

Some processes may take significant time. WFM Server has separate methods for initiating a long process, canceling the process, retrieving process status, retrieving process progress, and retrieving results.

Asynchronous methods take the user parameters, start the process, and return an immediate response. Later, you can request WFM Server to send the status (progress) of the process. To retrieve the status and progress of the request, use the Session service.

Note: Check the progress of an asynchronous request by calling `GetRequestProgress.` Call `GetRequestResult` only after the returned progress is 100%.

Synchronous methods start the process and return the result after the process has been completed.

Note: The `SchInsert.java` code sample shows how to use an asynchronous request. See “Using the Code Samples” on page 19 for the location of the code samples and how to use them.

method CloseRequest

The Session service provides a common method to close already-opened asynchronous requests. Asynchronous requests are opened by other services, using methods specific to the service. After an asynchronous request is complete, WFM Server stores the results, which can be retrieved using the request ID. The results are discarded only after `closeRequest` is called. If the request ID is not valid, this method returns an error.

Session Service Code Examples

The `SessionConnect.java` code sample shows how to open your session using the Locator and Session services, and how to close your session. See “Using the Code Samples” on page 19 for the location of the code samples and how to use them.

To run the sample, you must change the following variables to the correct ones for your environment:

- The WFM Server host name and port number.
- The `Application` object name of your client application as configured in Configuration Manager.

In addition to the `SessionConnect.java` code sample, a C# code example is included in this section, which shows how to establish a session.

Note: Like the code samples installed into the wFM Server directory from the Workforce Management CD, the following code example is intended for use as a guideline for building your own application. It is as error-free as possible, but it is not guaranteed or supported by Genesys.

C# Example

```
using System;
using System.Runtime.Remoting.Messaging;
using System.Runtime.Remoting.Metadata;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
using System.Collections;
using WFMLocatorService;
using WFMSessionService;
using WFMConfigService;

namespace WFMSessionClient
{
   class Example
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main(string[] args)
      {
         try
         {
            WFMLocatorServiceSoap locatorService = new WFMLocatorServiceSoap();
            locatorService.Url = "http://<WFM Server host name>:<WFM Server port
   number>/WFMServer?Handler=WFMLocatorService";
            LoginInfo loginInfo = locatorService.LocateServerLogin("<the Application object
   name of your client application as configured in Configuration Manager>", "default",
   "password");
            int siteID = loginInfo.siteID;

            WFMSessionServiceSoap sessionService = new WFMSessionServiceSoap();
            sessionService.Url = loginInfo.sessionServiceURL;
            SessionInfo sessionInfo = sessionService.OpenSession("<the Application object
   name of your client application as configured in Configuration Manager>", 0
   /*loginInfo.userID*/);

            String sid = sessionInfo.sid;

            Hashtable serviceMap = new Hashtable();

            foreach (ServiceInfo serviceInfo in sessionInfo.services)
            {
               serviceMap.Add(serviceInfo.serviceName, serviceInfo);
            }

            WFMConfigServiceSoap cfgService = new WFMConfigService.WFMConfigServiceSoap();
            cfgService.Url = ((ServiceInfo)serviceMap["ConfigService"]).serviceURL;

            CfgSiteHolder siteHolder = cfgService.GetSite(...);

            ...

            sessionService.CloseSession(sid);
         }
         catch (System.Runtime.Remoting.RemotingException e)
         {
           Console.WriteLine(e.Message);
        }
        catch (System.Net.WebException e)
        {
           Console.WriteLine("{0}, {1}", e.Message, " ", e.Response);
        }
        catch (System.Runtime.Remoting.ServerException e)
        {
           Console.WriteLine(e.Message);
        }
        catch (System.Runtime.Serialization.SerializationException e)
        {
           Console.WriteLine(e.Message);
        }
        catch (System.Exception e)
        {
           Console.WriteLine(e.Message);
        }
      }
   }
}
```

Util Essentials

The Util service has only a single class that you need to use, `OleDateTime`.

The `OleDateTime` class is used in the WFM API to represent a date and/or time value with no time zone or daylight saving time information attached.

Note: The code snippet below shows one use of `OleDateTime`. However, additional examples of its use occur throughout the code samples. See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

The following code snippet shows an example of the use of `OleDateTime`:

```
System.out.print(", Hire Date: ");
              System.out.print(new 
OleDateTime(cfgAgent.getWmStartDate()).toDateString());

              OleDateTime termDate = new 
OleDateTime(cfgAgent.getWmEndDate());
              if (termDate.getStatus() != OleDateTime.STATUS_NULL)
              {
                System.out.print(", Termination Date: ");
                System.out.print(termDate.toDateString());
              }
```

Chapter

3 Configuration

This chapter explains the Configuration service, represented by the `WFMConfigService7<xx>Soap` interfaces, which retrieve and update certain Workforce Management (WFM) configuration information.

This chapter includes the following sections:

Configuration Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15 and Chapter 2, “Locator, Session, and Util,” on page 23.

Configuration Essentials

The Configuration service is presented through the `WFMConfigService7<xx>Soap` interfaces. Use this service to retrieve information about configuration objects and to update some of these objects.

All the configuration and policies data from the WFM database is represented by configuration objects. Every object represents some logical entity and caches corresponding information from one ore more database tables.

These are main things you can do with the Configuration service:

- Retrieve configuration data about activities, agents, teams, Genesys states, business units, exception types, sites, skills, time-off types, time zones, users, and multi-site activities.
- Delete any of the objects listed in the previous list item.
- Set time zones.
- Update activities and agents.

Note: For descriptions of all the configuration objects stored in the WFM database, see *Workforce Management 7.5 Configuration Utility Help*. You can open the Help file from the WFM Configuration Utility by clicking Help or access a stand-alone version of the Help file on the Genesys Technical Support website.

Retrieving Configuration Data

To retrieve configuration data, you can call the appropriate method for the object you want. For example, to retrieve time zone data, use the `getTimezone` method.

The `CfgTest.java` code example shows how to retrieve many configuration object types. See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

The following code short code example shows how to retrieve all time zones:

Time Zones

```
// Retrieve all Time Zones configured
      CfgTimezoneHolder cfgTimezoneHolder = cfgService.getTimezone(
          null, // want all time zones
          ECfgSortMode.Timezone.CFG_TIMEZONE_SORT_NONE,
          true,
          ECfgInfoType.CFG_INFO_OBJECT_SHORT
          );

      // Store all time zones in a map so I could easily find them 
later by ID
      Hashtable timezoneMap = new Hashtable();
      for (int i = 0; i < 
cfgTimezoneHolder.getObjectShortArrayNSizeIs(); ++i)
      {
        CfgTimezoneShort cfgTimezone = 
cfgTimezoneHolder.getObjectShortArray()[i];
        timezoneMap.put(new Integer(cfgTimezone.getWmTimezoneId()), 
cfgTimezone);
      }
```

Setting Time Zones

The `setLocalTimezone` method enables you to change the default time zone used in schedule, performance, and adherence data.

Updating Configuration Objects

The `CfgUpdate.java` code sample shows how to use the `updateAgent` and `updateActivity` methods to change configuration information about Agent and Activity objects.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Chapter

4 Calendar

This chapter explains the Calendar service, represented by the WFMCalendarService7<xx>Soap interfaces, which retrieve Calendar-item data and enable you to update certain items.

This chapter includes the following sections:

Calendar Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15 and Chapter 2, “Locator, Session, and Util,” on page 23.

Calendar Essentials

The Calendar service is presented through the WFMCalendarService7<xx>Soap interfaces. Use this service to work with Calendar-item data, such as exceptions, time off, availability, rotating patterns (read-only), days off, shifts, and working hours.

These are the main things you can do with the Calendar service:

- Retrieve, insert, edit, and delete Calendar items.
- Change the status (preferred, granted, or declined) of calendar items.

- Calculate time-off balances and carry-over information.

A major feature of the Calendar service is priority resolution between calendar items. After the priority-resolution rules are applied, each Calendar item for a day has a resolved status and can be correctly scheduled.

The Calendar provides information about agent time off, such as accrued hours, bonus hours, and hours taken in advance. The Calendar can also calculate an agent's time-off balance for any selected period.

You can also use the Calendar to calculate the carry-over for any time-off type.

Note: For more information about Calendar items, such as availability patterns and availability preferences, see *Workforce Management 7.5 Web for Supervisors Help*. You can open it from any WFM Web for Supervisors window or you can use the stand-alone version, located on the Genesys Technical Support website.

Time Off Balances and Carry-Over

The Calendar's time off functionality does the following:

- Calculates accrued and awarded time-off hours based on the configured time-off rule.
- Calculates the time-off balance (in hours) for a requested date or selected period.
- Stores data about bonus time off and time off taken in advance.
- Calculates the carry-over date based on the carry-over day set using the WFM Configuration Utility, and tracks the carried-over hours for each time-off type.

When you specify a date (or period), the Calendar calculates the time-off balance for that date or period. All scheduled, granted, and preferred time off is subtracted from the time-off balance. Bonus hours can be added to the time-off balance.

You can also retrieve bonus, advance, and carried-over hours.

Agents accrue each type of time off as specified in the agent's time-off rule for that time-off type. A limited number of time-off hours can be carried over to the next time-off period in addition to the maximum number of time-off hours allowed.

Working with Calendar Items

The `CalTest.java` code example shows how to insert agent Calendar items for a specified date, retrieve Calendar data for the site on that date, and delete previously-entered agent Calendar items.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Chapter

5 Forecast

This chapter explains the Forecast service, represented by the `WFMForecastService7<xx>Soap` interfaces, which retrieve forecast data.

This chapter includes the following sections:

Forecast Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15 and Chapter 2, “Locator, Session, and Util,” on page 23.

Forecast Essentials

The Forecast service is presented through the `WFMForecastService7<xx>Soap` interfaces. Use this service to retrieve forecast information.

Retrieving Forecast Data

Use the `getForecastInformation` method to retrieve forecast data from a scenario, or from the Master Forecast, for a specified period of time.

Note: Forecast data is read-only. You cannot create or modify forecasts using the WFM Integration API.

The Forecast service can retrieve forecast scenario or Master Forecast data for the following levels, listed from the most limited to the widest-ranging:

- Activity
- Multi-site Activity
- Business Unit
- Site
- Enterprise

The Forecast service accepts one level per request. For example, you can request data for one site or one activity.

The Forecast service retrieves the following data categories:

- Interaction Volume (IV)
- Average Handling Time (AHT)
- Staffing requirements

If you request data for more than one activity, aggregation formulas are used to present total value for the level you select. The Forecast service can access all available Master Forecast and forecast scenario data. There is no restriction on the date range you can request.

The Forecast service supports the following granularities:

- 15, 30, and 60 minutes
- daily, weekly, and monthly

The Forecast services also uses aggregation formula to return data with a granularity greater than 15 minutes.

Data for the Master Forecast and forecast scenarios has the same structure, although scenarios have some fields that do not exist in the Master Forecast. To view Master Forecast data, use the `getForecastInformation` method, and specify `0` as the value for the `scenarioID` parameter.

Staffing Forecasts

Staffing requirements indicate the number of agents required to adequately staff the contact center, within defined service objectives and parameters. Staffing requirements created by the Forecast service are used by WFM Builder to build more accurate schedules.

Staffing requirements stored for each Activity in WFM Database. Staffing data is included in scenarios or the Master Forecast, together with the IV and AHT data. You can retrieve staffing forecast data for the same levels and granularities as for IV and AHT.

The `FrcTest.java` code sample shows how to retrieve the desired data.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Chapter

6 Schedule

This chapter explains the Schedule service, represented by the `WFMScheduleService7<xx>Soap` interfaces, which retrieve schedule data and enable you to update certain items.

This chapter includes the following sections:

Schedule Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15 and Chapter 2, “Locator, Session, and Util,” on page 23.

Schedule Essentials

The Schedule service is presented through the `WFMScheduleService7<xx>Soap` interfaces. Use this service to retrieve and update schedule information.

These are the main things you can do with the Schedule service:

- Retrieve agent schedule data, organized by agent or by type of schedule item.
- Insert an item into or delete one from an agent schedule day.

- Retrieve totals for various items for a specified period of time (for example, total time off or total paid hours). You can retrieve totals by site, team, or agent.

You can retrieve the following read-only schedule information for the requested period:

- Agent
- Agent's schedule state

Typically, the agent schedule state includes the following information:

- Schedule state type
- Schedule state ID (including Activity ID if the state is a work activity)
- Schedule state name
- Schedule state start time
- Schedule state end time
- Whether the schedule state is paid
- Whether the schedule state is full day

The `SchState.java` code sample shows how to retrieve schedule-state data.

See “Using the Code Samples” on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Agent Scheduled States

The Scheduled state is the most basic element of an agent’s schedule. The agent schedule is built on a daily basis, and every day schedule contains one or more states scheduled. At any given moment during the day, an agent is assigned one scheduled state (in most cases). However, multi-skilled agents can be scheduled to work on more than one activity at the same time. In such cases an agent may have more than one scheduled state for a given period of time. Usually, scheduled states have defined start and end times, but some states are full-day, and have no start or end time defined. The following tables describe various characteristics of scheduled states.

- Table 1, “Scheduled State Types,” on page 43.
- Table 2, “Special Scheduled State Types,” on page 44.
- Table 3, “Agent Scheduled State Requests,” on page 44
- Table 4, “Hard Schedule State Editing Constraints,” on page 45
- Table 5, “Soft Schedule State Editing Constraints,” on page 46

Table 1 describes Scheduled state types.

Table 1: Scheduled State Types

Scheduled State	Is Full-Day	Description
Day Off	Yes	The `Day Off` state is a full-day state.
Time Off	Yes/No	The `Time Off` state may or may not be a full-day state, depending on the type of time off.
Exception	Yes/No	The `Exception` state may or may not be a full-day state, depending on the exception type.
Break	No	The `Break` state is a part-day state, and, as such, always has a defined start and end time. This state can only exist within shift boundaries.
Meal	No	The `Meal` state is a part-day state, and, as such, always has a defined start and end time. This state can only exist within shift boundaries.
Activity	No	The `Activity` state is a part-day state, and, as such, always has a defined start and end time. Since multi-skill agents can be scheduled to work on more than one activity at a time, there can be several overlapping `Activity` states for the same period of time. This state can only exist within shift boundaries.

Special Scheduled State Types

Table 2 on page 44 notes special Schedule state types. These differ from standard ones in that they are categories, or groupings, used for convenience rather than actual work activities or other schedule items, such as breaks and meetings, that agents actually perform.

Table 2: Special Scheduled State Types

Scheduled State	Is Full-Day	Description
Activity Set	No	Agents are never scheduled for this state. This state is available as a means to group several `Activity` states into one logical state (as an `Activity Set` groups several `Activities`). Note: all `Activity` states that are not assigned to a particular `Activity Set,` are assigned to an unnamed `Activity Set` state.
Shift	No	Agents are never scheduled for this state. It is available to describe scheduled shift boundaries into which part-day states can fit. There may also be a case where, because of constraints, no state can be scheduled for some time window within the shift boundaries. In that case, the `Shift` state is provided as the scheduled state for that time, and it should be treated as a forced paid break.
Marked Time	No	This state is used to mark a period of time within a shift.

Scheduled State Information

Table 3 on page 44 describes the different components of an agent scheduled state request:

Table 3: Agent Scheduled State Requests

Scheduled State	Description
Scheduled State Type	Identifies the type of scheduled state.
State ID	Identifies the specific scheduled state, based on `Scheduled State Type` (for instance, `Meal` or `Break ID, Exception ID,` or `Activity ID`).
State Start Time	Scheduled state start time (undefined if state is full-day)

Table 3: Agent Scheduled State Requests (Continued)

Scheduled State	Description
State End Time	Scheduled state end time (undefined if state is full-day)
Is Paid?	Identifies if a state is paid or unpaid.

Schedule States Editing Constraints

Any agent schedule day that contains a Scheduled state with a hard constraint violated is considered inconsistent and is rejected by the Schedule service. Violation of soft constraints generates warnings, but you can add the item anyway. When you are inserting schedule items, the constraints shown in Table 4 (hard constraints) and Table 5 on page 46 (soft constraints) are applied.

Table 4: Hard Schedule State Editing Constraints

State Type	Hard Constraints
Break	For both these states: • The start and end times should not overlap any other scheduled state that has the same “weight”—any other `Break` or `Meal` states. • Must be positioned within shift state start and end times. These states always have shift state underneath. • The minimum duration for these states is `1` minute.
Meal	
Part-Day Exception	• The start and end times should not overlap any other scheduled state that has the same “weight”—any other `Part-Day Exception` state. • Must be positioned within shift state start and end times. This state always has shift state underneath. • The minimum duration for this states is `1` minute.
Full-Day Exception	Must be the only scheduled state in a given day.

Table 4: Hard Schedule State Editing Constraints (Continued)

State Type	Hard Constraints
Activity Set	• The start and end times should not overlap any other scheduled state that has the same “weight”—any other `Activity Set` state. • Must be positioned within shift state start and end times. This state always has shift state underneath. • A connection point for one `Activity Set` with another should always be at a 15-minute alignment. • For each distinct `Activity Set` in a list of IDs for `Activities` in a given `Agent Day,` there must be at least one `Activity Set` state.
Activity	Read Only. Activity states passed to WFM Server are ignored. The Schedule Service only attends to the list of `Activity` identifiers in an `Agent Day.` (It is possible to add or remove scheduled `Activities` by modifying this list.)

Table 5: Soft Schedule State Editing Constraints

State Type	Soft Constraints

Break	`MinLengthBefore`
	`MaxLengthBefore`
	`MinLengthAfter`
	`StartStep`
	`StartOffset`
	`MaxDistance`
	`Anchor`
	`Duration`
	`Paid`

Meal	`StartTime`
	`EndDuration`
	`Duration`
	`LengthBeforeMeal`
	`LengthAfterMeal`
	`StartStep`
	`Paid`

Table 5: Soft Schedule State Editing Constraints (Continued)

State Type	Soft Constraints

Activity Set	`MinDuration`
	`Strict`
	`Open Hours`

Shift	`MinDuration`
	`MaxDuration`
	`EarliestTimeStart`
	`LatestEndDuration`
	`Open Week Days`
	`StartStep`
	`BreakMinDistance`
	`Shift Meals`
	`Shift Breaks`
	`Agent Open Hours`

Day Off	Minimum/maximum days off per week/planning period
Time Off	See `Calendar` service for more information.
Note: All states have minimum/maximum paid hours per day/week/planning period.	

Retrieving Agent-Day Schedule Data

An agent-day schedule, or `Agent Day,` includes the agent's schedule for one day. In this case, *day* does not mean a calendar day, but instead a *shift day*. That is, if an agent's shift starts today and continues into tomorrow, all the agent's scheduled states, including the ones that extend into the second calendar day, are still assigned to today's agent-day schedule.

However, no scheduled states may extend past noon of the second day. As a result, an agent day is effectively 36 hours.

Table 6 on page 48 shows the information that is included in the `Agent Day`.

Table 6: Agent Schedule Day Information

Attributes	Description	
Agent ID	Identifies the agent for whom the schedule is built.	
Site ID	The agent's site **Note:** schedules are bound to sites. So, though an agent can change sites after a schedule is built, the schedule remains associated with the agent's original site.	
DayDate	The date of the `Agent Day` schedule	
Agent Day Type	Agent Day Types	
	Day Off	The `Agent Day` schedule is a day off.
	Time Off	The `Agent Day` schedule is time off.
	Exception	The `Agent Day` schedule is a full-day exception.
	Shift	The `Agent Day` schedule is a shift.
	Shift-Exception	The `Agent Day` schedule is one or more non-full-day exceptions, with defined start and end times, and there is no shift scheduled.
Day Start Time	Start time of the first scheduled state in the `Agent Day` schedule.	
Day End Time	End time of the first scheduled state in the `Agent Day` schedule.	
Shift ID	Identifies scheduled shift, if any.	
List of Scheduled Activities	A list of identifiers for `Activities` on which the `Agent Day` schedule was built.	
Scheduled States	A list of all scheduled states in the `Agent Day` schedule. (The states can be presented in a sequential or overlapping fashion.)	
Activity Coverage	The Activity coverage distribution within the scheduled `Activity` states for every 15-minute interval. (The Schedule Builder creates this important of an `Agent Schedule Day.`)	

Inserting Schedule Items

You can insert, edit and delete schedule items.

If you are inserting a full-day schedule item, start time and end time are ignored, unless the item is a full-day exception. If it is full-day exception, the difference in minutes between the end time and the start time represents the paid duration for the full-day exception. If the difference is zero, then the standard working hours are used.

This operation is asynchronous. When you send you initial request, WFM Server returns a requestID. Use the Session service to check the progress of the asynchronous request. Use the `getRequestValidation` method to retrieve results when the process is complete.

The `SchInsert.java` code example shows how to insert an item. Editing and deleting are performed in much the same way.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Retrieving Totals

You can retrieve schedule information about specific agents for each day in a specified time period. The `getAgentTotals` function can be used to calculate daily, weekly totals, monthly totals, schedule planning period, or yearly totals for the following:

- Total paid hours
- Total work hours
- Total hours in specified scheduled states
- Total hours in specified marked time types
- Total overtime hours

The `SchTotals.java` code example shows how to retrieve this data.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Chapter

7 Schedule Trading

This chapter explains the Schedule Trading service, represented by the `WFMScheduleTradeService7<xx>Soap` interfaces, which retrieve schedule-trading data, and enable you to approve or deny trades.

This chapter includes the following sections:

Schedule Trading Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15 and Chapter 2, “Locator, Session, and Util,” on page 23.

Schedule Trading Essentials

The Schedule Trading service is presented through the `WFMScheduleTradingService7<xx>Soap` interfaces. Use this service to retrieve and update schedule-trading information.

As a supervisor, the main things you can do with the Schedule Trading service are the following:

- Retrieve data about proposed schedule trades.
- Approve trade proposals.

As an agent, the main things you can do with the Schedule Trading service are the following:

- Review schedule trade proposals.
- Accept, decline, or cancel trade proposals.
- Create new trade proposals.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

There are two types of trade proposal, community and personal. A community proposal is open to any agent with a compatible shift. A personal proposal is directed to a specific individual.

Figure 2 on page 53 shows the interaction flow for a community trade proposal. Figure 3 on page 54 shows the interaction flow for a personal trade proposal.

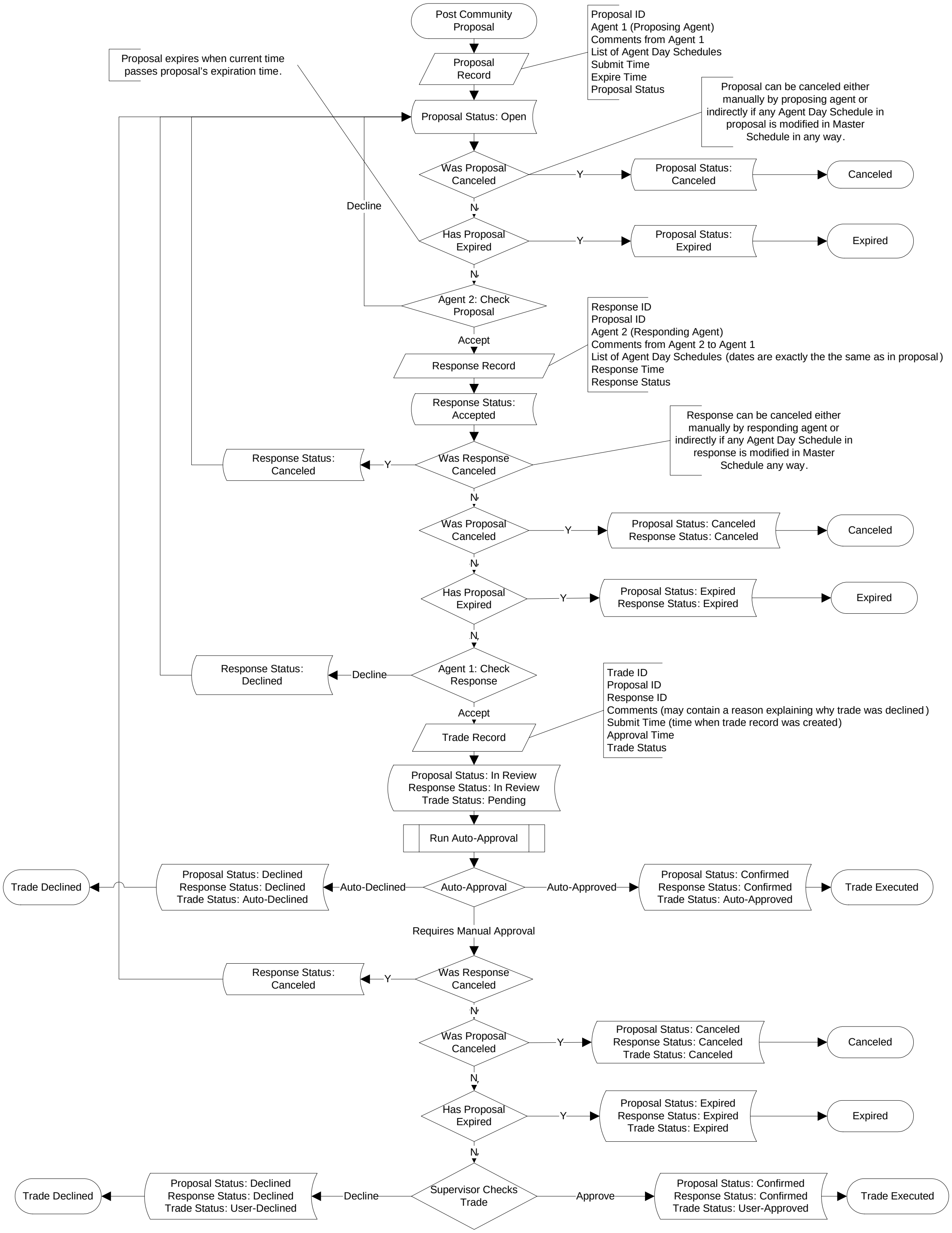
Post Community Proposal
Proposal Record
Proposal ID
Agent 1 (Proposing Agent)
Comments from Agent 1
List of Agent Day Schedules
Submit Time
Expire Time
Proposal Status
Proposal expires when current time passes proposal's expiration time.
Proposal Status: Open
Proposal can be canceled either manually by proposing agent or indirectly if any Agent Day Schedule in proposal is modified in Master Schedule in any way.
Was Proposal Canceled
Y
Proposal Status: Canceled
Canceled
N
Decline
Has Proposal Expired
Y
Proposal Status: Expired
Expired
N
Agent 2: Check Proposal
Accept
Response Record
Response ID
Proposal ID
Agent 2 (Responding Agent)
Comments from Agent 2 to Agent 1
List of Agent Day Schedules (dates are exactly the the same as in proposal)
Response Time
Response Status
Response Status: Accepted
Response can be canceled either manually by responding agent or indirectly if any Agent Day Schedule in response is modified in Master Schedule any way.
Response Status: Canceled
Y
Was Response Canceled
N
Was Proposal Canceled
Y
Proposal Status: Canceled
Response Status: Canceled
Canceled
N
Has Proposal Expired
Y
Proposal Status: Expired
Response Status: Expired
Expired
N
Response Status: Declined
Decline
Agent 1: Check Response
Accept
Trade Record
Trade ID
Proposal ID
Response ID
Comments (may contain a reason explaining why trade was declined)
Submit Time (time when trade record was created)
Approval Time
Trade Status
Proposal Status: In Review
Response Status: In Review
Trade Status: Pending
Run Auto-Approval
Trade Declined
Proposal Status: Declined
Response Status: Declined
Trade Status: Auto-Declined
Auto-Declined
Auto-Approval
Auto-Approved
Proposal Status: Confirmed
Response Status: Confirmed
Trade Status: Auto-Approved
Trade Executed
Requires Manual Approval
Response Status: Canceled
Y
Was Response Canceled
N
Was Proposal Canceled
Y
Proposal Status: Canceled
Response Status: Canceled
Trade Status: Canceled
Canceled
N
Has Proposal Expired
Y
Proposal Status: Expired
Response Status: Expired
Trade Status: Expired
Expired
N
Trade Declined
Proposal Status: Declined
Response Status: Declined
Trade Status: User-Declined
Decline
Supervisor Checks Trade
Approve
Proposal Status: Confirmed
Response Status: Confirmed
Trade Status: User-Approved
Trade Executed

Figure 2: Community Trade Proposal Flow Diagram

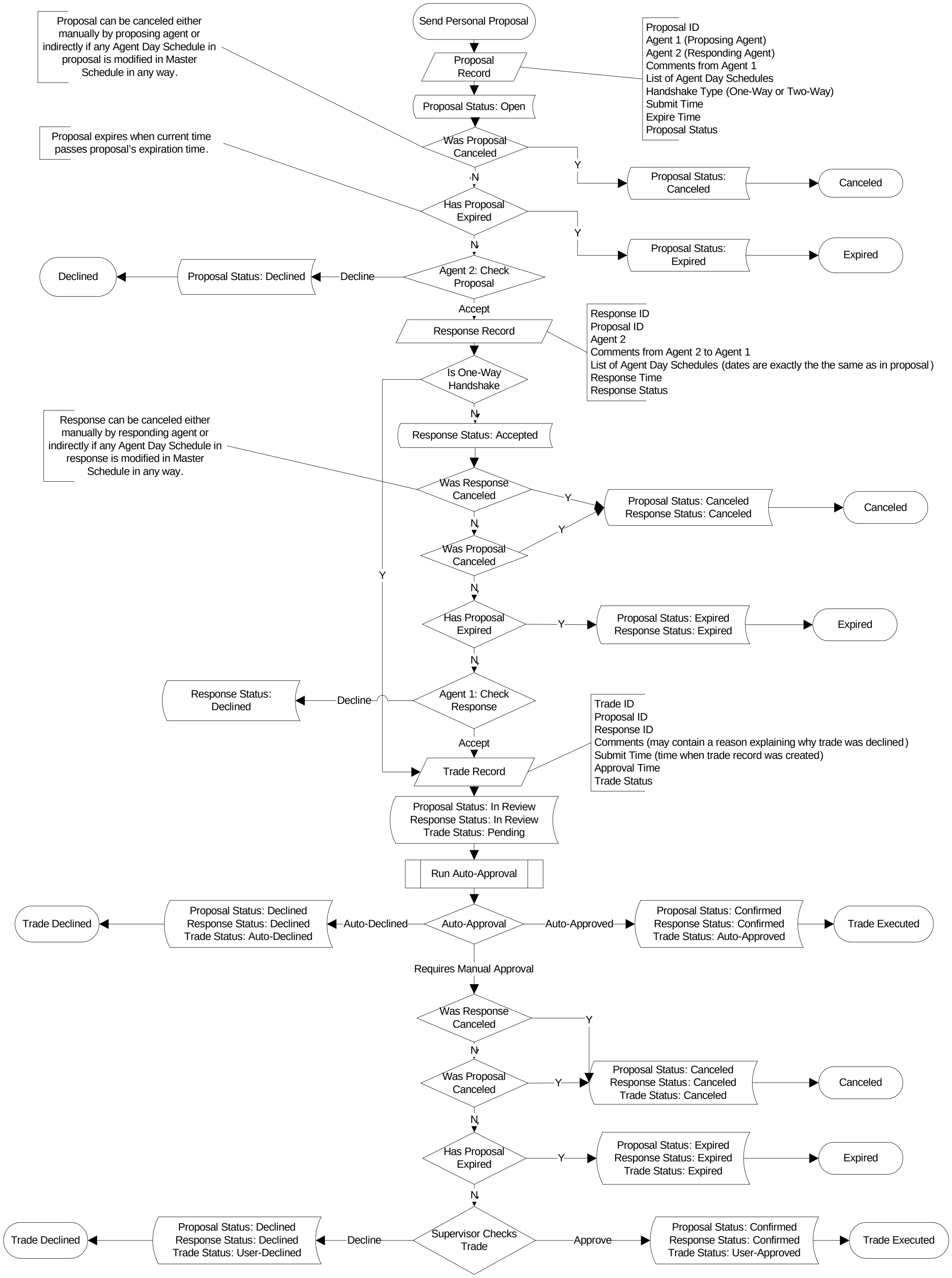
Send Personal Proposal
Proposal can be canceled either manually by proposing agent or indirectly if any Agent Day Schedule in proposal is modified in Master Schedule in any way.
Proposal Record
Proposal ID
Agent 1 (Proposing Agent)
Agent 2 (Responding Agent)
Comments from Agent 1
List of Agent Day Schedules
Handshake Type (One-Way or Two-Way)
Submit Time
Expire Time
Proposal Status
Proposal Status: Open
Proposal expires when current time passes proposal's expiration time.
Was Proposal Canceled
Y
Proposal Status: Canceled
Canceled
N
Has Proposal Expired
Y
Proposal Status: Expired
Expired
N
Declined
Proposal Status: Declined
Decline
Agent 2: Check Proposal
Accept
Response Record
Response ID
Proposal ID
Agent 2
Comments from Agent 2 to Agent 1
List of Agent Day Schedules (dates are exactly the the same as in proposal)
Response Time
Response Status
Is One-Way Handshake
N
Response Status: Accepted
Response can be canceled either manually by responding agent or indirectly if any Agent Day Schedule in response is modified in Master Schedule in any way.
Was Response Canceled
Y
Proposal Status: Canceled
Response Status: Canceled
Canceled
N
Was Proposal Canceled
Y
Y
N
Has Proposal Expired
Y
Proposal Status: Expired
Response Status: Expired
Expired
N
Response Status: Declined
Decline
Agent 1: Check Response
Accept
Trade Record
Trade ID
Proposal ID
Response ID
Comments (may contain a reason explaining why trade was declined)
Submit Time (time when trade record was created)
Approval Time
Trade Status
Proposal Status: In Review
Response Status: In Review
Trade Status: Pending
Run Auto-Approval
Trade Declined
Proposal Status: Declined
Response Status: Declined
Trade Status: Auto-Declined
Auto-Declined
Auto-Approval
Auto-Approved
Proposal Status: Confirmed
Response Status: Confirmed
Trade Status: Auto-Approved
Trade Executed
Requires Manual Approval
Was Response Canceled
Y
N
Was Proposal Canceled
Y
Proposal Status: Canceled
Response Status: Canceled
Trade Status: Canceled
Canceled
N
Has Proposal Expired
Y
Proposal Status: Expired
Response Status: Expired
Trade Status: Expired
Expired
N
Trade Declined
Proposal Status: Declined
Response Status: Declined
Trade Status: User-Declined
Decline
Supervisor Checks Trade
Approve
Proposal Status: Confirmed
Response Status: Confirmed
Trade Status: User-Approved
Trade Executed

Figure 3: Personal Trade Proposal Flow Diagram

Trade Proposals

Table 7 shows trade proposal parameters.

Table 7: Trade Proposal Parameters

<table>
<tr><th>Attribute</th><th colspan="2">Description</th></tr>
<tr><td>Proposal ID</td><td colspan="2">Identifies the agent for whom the schedule is built.</td></tr>
<tr><td>Proposing Agent</td><td colspan="2">Proposing agent.</td></tr>
<tr><td>Site</td><td colspan="2">The agent’s site.</td></tr>
<tr><td>Schedule Data</td><td colspan="2">A list of <code>Agent Day</code> schedules proposed for the trade.</td></tr>
<tr><td>Comments</td><td colspan="2">Any comments from the proposing agent.</td></tr>
<tr><td>Submit Time</td><td colspan="2">A submit date and time (in the site’s time zone).</td></tr>
<tr><td>Expiration Time</td><td colspan="2">Proposal’s expiration date and time (in the site’s time zone). <b>Note:</b> the expiration time should be earlier than the first date in the proposed schedules for trading.</td></tr>
<tr><td rowspan="3">Proposal Type</td><td colspan="2">Proposal Type and Description</td></tr>
<tr><td>Personal</td><td>The proposal is sent directly to the agent.</td></tr>
<tr><td>Community</td><td>The proposal is posted to the agent community so that anyone may respond.</td></tr>
<tr><td rowspan="3">Handshake Type</td><td colspan="2">Handshake Type and Description</td></tr>
<tr><td>One-Way</td><td>For <code>Personal</code> proposals only: no response is required from the agent.</td></tr>
<tr><td>Two-Way</td><td>Any response to the proposal requires approval from the proposing agent.</td></tr>
</table>

Table 7: Trade Proposal Parameters (Continued)

Attribute	Description	
Proposal Status	Proposal Status and Comments	
	Open	The proposal has posted to the community (a `Community` proposal), or has been sent directly to the agent (a `Personal` proposal).
	In Review	The proposal has been accepted by both parties, but the trade was not auto-approved, and is awaiting approval by the supervisor.
	Confirmed	The proposal was accepted by both parties, has been approved (automatically or manually), and the trade has been executed.
	Declined	The proposal was declined by either the supervisor or the responding agent (if the proposal was `Personal`).
	Canceled	The proposal was cancelled by the proposing agent or was indirectly cancelled because of a change to the schedule referred to in the proposal.
	Expired	The current time has now passed the proposal's `Expiration Time.`

Trade Responses

Table 8 shows parameters included in a trade response.

Table 8: Trade Response Parameters

Attribute	Description
Response ID	A unique ID for the response.
Proposal ID	The ID of the proposal related to this response.
Responding Agent	The proposing agent.
Comments	Any comments from the responding agent.

Table 8: Trade Response Parameters (Continued)

Attribute	Description	
Schedule Data	A list of `Agent Day` schedules that are a response to the proposed schedules. The dates of the schedules in the response are an exact match of the schedules in the proposal.	
Response Time	The response date and time (in the site's time zone).	
Response Status	Response Status and Comments	
	Accepted	A responding agent has accepted a proposal.
	In Review	A proposal has been accepted by both parties, but the trade was not auto-approved, and is awaiting manual approval by the supervisor.
	Confirmed	A proposal was accepted by both parties, and the trade has been both approved (automatically or manually), and executed.
	Declined	A response was declined by either the supervisor or the proposing agent (if the proposal was `Personal`).
	Canceled	The response was cancelled by the responding agent or the proposal was cancelled by the proposing agent (and so the response is cancelled as well). It may have been indirectly cancelled because of a change to the schedule referred to in the proposal or the response.
	Expired	The current time has now passed the proposal's `Expiration Time.`

Trade Record

The trade record is a summary of the a trade, including its ID and its status. Table 9 shows data included in a trade record.

Table 9: Trade Record Parameters

Attribute	Description	
Trade ID	A unique ID for the trade.	
Proposal ID	The ID of the proposal related to this trade.	
Response ID	The ID of the response related to this trade.	
Comments	May contain a reason why the trade was declined.	
Submit Time	Date and time for the creation of the trade record.	
Approval Time	Time of the approval, if appropriate.	
Approving User	Approving user name (if the trade was approved).	
Trade Status	Trade Status and Comments	
	Pending	The trade could not be auto-approved and is now awaiting manual approval by the supervisor.
	Auto-Declined	The system auto-declined the trade.
	User-Declined	The supervisor manually declined the trade.
	Auto-Approved	The system auto-approved the trade.
	User-Approved	The supervisor manually approved the trade.
	Cancelled	Either the proposing or responding agent manually cancelled the trade, or the a modification in the master schedule of any of the Agent Day schedules participating in the trade caused its indirect cancellation.

Agent Trading—Proposals

There are a number of actions that an agent can take when initiating a trade proposal.

Post Community Trade Proposal

A trade proposal is posted to community and is available to all qualifying agents.

The following information is provided by the proposing agent:

- Proposing agent
- List of dates for which agent wants to trade schedules
- Comments about the proposal (optional)
- Proposal expiration date. An expiration date must be earlier than the first day in a list of days proposed for trade. The default value is one day earlier than the first day in proposal.

API function: `PostTradeProposal`

Send Personal Trade Proposal

A trade proposal is sent directly to specified agent. Personal proposal can have one-way or two-way handshake assigned. One-way handshake does not require a response from responding agent. It assumes that proposing agent is aware of responding agent's schedule and is ready to trade upon proposal's acceptation by responding agent. Two-way handshake requires acceptation of response by proposing agent after proposal is accepted by responding agent.

The following information is provided by proposing agent:

- Proposing agent
- Responding agent (An agent to whom proposal is addressed to)
- Handshake type (One-way or Two-way)
- List of dates for which agent wants to trade schedules
- Comments from proposing agent addressed to responding agent (optional)
- Proposal expiration date. An expiration date must be earlier than the first day in a list of days proposed for trade. The default value is one day earlier than the first day in proposal.

An API function: `SendPersonalTradeProposal`

Cancel Trade Proposal

A proposal can be canceled by proposing agent at any time while proposal is in Open or In Review status. If proposal is canceled, all responses to this proposal are automatically canceled as well. If there is trade pending for this proposal, it is also canceled.

The following information is provided by proposing agent:

- Proposal ID
- Comments explaining why proposal was canceled (optional). These comments are appended to the comments field in proposal record and response record (if there is any).

API function: `CancelTradeProposal`

Decline Response to Trade Proposal

A proposing agent can select to decline response from community to community proposal as well as response from agent responding to personal proposal if two-way handshake was requested. Once accepted, response cannot be declined by proposing agent.

The following information is provided by proposing agent:

- Response ID
- Comments explaining why response was declined (optional). These comments are appended to the comments field in response record.

API function: `DeclineTradeResponse`

Accept Response to Trade Proposal

A response to proposal should be accepted by proposing agent in order to schedule trade to be submitted for approval. An exception to this is personal proposal with one-way handshake. In that case trade is submitted upon acceptation by responding agent.

The following information is provided by proposing agent:

- Proposal ID
- Response ID
- Comments addressed to supervisor (optional). These comments are written to the comments field in trade record.

API function: `AcceptTradeResponse`

Agent Trading—Responses

The following section explains the actions an agent can take in response to a trade proposal.

Accept Trade Proposal

Acceptation of proposal creates a response record, which goes back to proposing agent for final handshake. An exception to this is personal proposal with one-way handshake. In that case acceptation of proposal by responding agent automatically submits a trade record for approval.

The following information is provided by responding agent:

- Responding agent
- Proposal ID
- Comments addressed to proposing agent. These comments are written to the comments field in response record.

API function: `AcceptTradeProposal`

Decline Personal Trade Proposal

Responding agent may select to decline personal trade proposal. Community trade proposal cannot be declined by responding agent. Once accepted, personal trade proposal cannot be declined.

The following information is provided by responding agent:

- Responding agent
- Proposal ID
- Comments addressed to proposing agent

API function: `DeclinePersonalTradeProposal`

Cancel Trade Response

A response to proposal can be canceled by responding agent at any time while proposal is in Accept or In Review status. If response is canceled, a corresponding proposal is re-listed (reset to Open status) and pending trade (if any) is canceled.

The following information is provided by responding agent:

- Response ID
- Comments explaining why response was canceled (optional). These comments are appended to the comments field in response record.

API function: `CancelTradeResponse`

Trading—Supervisor Actions

It is the supervisor's responsibility to review pending trade requests and approve or decline them in timely fashion. If supervisor selects to approve pending trade, it doesn't necessarily mean that trade will be approved. The auto-decline conditions are re-checked again and if there were any changes to agent's profile that would prevent schedule trade then trade would be auto-declined instead.

Approve Trade

The following information is provided by supervisor

- Trade ID
- Comments (optional). These comments are appended to the comments field in trade, proposal and response records.

An API function: `ApproveTrade`

Decline Trade

The following information is provided by supervisor

- Trade ID
- Comments explaining why trade was declined (required). These comments are appended to the comments field in trade, proposal and response records. In case of auto-decline, a reason is generated automatically by the system.

API function: `DeclineTrade`

Auto-Approval

Auto-Approval process is run every time once trade record is submitted, that is after response is accepted by proposing agent or after personal proposal is accepted by responding agent in case of one-way handshake. Auto-approval procedure applies trading rules, as explained in the section below, and decides whether trade can be auto-approved, auto-declined or if it requires manual review by supervisor to decide its fate.

Trading Rules

Different schedule items have different rules for whether and how they can be traded.

Exception Trade Rules

All configured exception types have an Exception Trade Rule specified, as noted in Table 10. The Exception Trade Rule specifies what to do with scheduled exception in case of schedule trade.

Table 10: Trading Rule Exceptions

Exception Trade Rule	Comments
Delete	Delete exception from schedule and do swap.
Do not trade	Do not trade schedules.
Keep with agent	Swap traded schedules but keep scheduled exception with original agent. Vacation type exceptions will always have this flag or the `Delete` flag set.
Keep with schedule	Swap traded schedules together with exceptions.

Auto-Decline Conditions

The following are reasons that a trade cannot occur, even if a supervisor approves it.

- Traded schedules have scheduled work on activities that any of trading parties are not qualified to work on.
- Any of the trading parties have full-day time off scheduled.
- Any of the trading parties have scheduled exception(s) with Do not trade rule configured.
- Schedule swap would create inconsistent schedule. For instance, if one agent has exception with Keep with Agent trading rule and another has exception with Keep with Schedule trading rule and both exceptions have overlapping time. In such case schedule swap would create exceptions with overlapping time, which is inconsistent schedule.

Manual Approval Conditions

- Days off are traded for working days.
- Traded schedules have different work durations.
- Schedule trade violates agent availability (contract, granted or rotating availability).
- Schedule trade violates contract min/max working hours per day.
- Schedule trade violates contract min/max working hours per week.
- Schedule trade violates the day-to-day distance rule.
- Schedule trade violates contract min/max days off per week.
- Schedule trade involves exception with Delete trading rule.
- Keep-with-Agent exception is exchanged with exclusive work.

Auto-Approval Conditions

All schedule trades that do not fall in Auto-Decline and Manual Approval conditions are approved and traded schedules are swapped automatically.

Auto-Cancel

Pending trades, proposals or responses are automatically cancelled if proposal or response schedules are modified in Master Schedule in any way. For instance, that could be schedule modified manually by supervisor or another trade with overlapping schedule data was approved and executed.

Trade Queries

The Schedule Trade service allows you to query trade proposals addressed to or originating from a specified agent, responses, and trade records. It allows you to specify different filters and a sort order. The query result is a list, sorted according to the proposal, response, or trade records.

Querying Trade Proposals

The query to retrieve schedule-trade-proposal data must include the parameters shown in Table 11.

Table 11: Trade Proposal Query Parameters

Input Information	Comments	
Agent	The agent from whose perspective the proposals are queried.	
Origination Filter	Origination and Comments	
	My Proposals	Proposals posted by the specified agent.
	Other Proposals	Proposals posted by all other agents for which the specified agent is qualified.
	All Proposals	All proposals, regardless of origination.
Date Window Filter	Defines the date window into which the proposed schedule should fall.	
Proposal Type Filter	Proposal Type and Comments	
	Personal	`Personal` trade proposals.
	Community	`Community` trade proposals.
	Both	`Personal and Community` trade proposals.
Proposal Status Filter	Allows you to define a filter of one or several proposal statuses.	

Table 11: Trade Proposal Query Parameters (Continued)

Input Information	Comments	
Sort Order	Sort Order and Comments	
	Agent	Proposals sorted according to the proposing agent.
	Status	Sorted by `Status.`
	Type	Sorted by `Type.`
	Paid Hours	Sorted by the number of paid hours in the proposed schedules.
	Submit Time	Sorted by the time of submission.
	Expire Time	Sorted by the expiration time.

Querying Trade Responses

The query to retrieve schedule-trade-response data must include the parameters shown in Table 12.

Table 12: Trade Response Query Parameters

Input Information	Comments	
Agent	The agent from whose perspective the proposals are queried.	
Origination Filter	Origination and Comments	
	My Responses	Responses sent by the specified agent.
	Other Responses	Responses received by the specified agent.
	All Responses	All responses, regardless of origination.
Date Window Filter	Defines the date window into which the response schedule should fall.	
Response Status Filter	Allows you to define a filter of one or several response statuses.	

Table 12: Trade Response Query Parameters (Continued)

Input Information	Comments	
Sort Order	Sort Order and Comments	
	Agent	Responses sorted according to the responding agent.
	Status	Sorted by `Status.`
	Paid Hours	Sorted by the number of paid hours in the proposed schedules.
	Response Time	Sorted by the time of the response.
	Expire Time	Sorted by the response expiration time.

Querying Trades

The query to retrieve schedule-trade data must include the parameters shown in Table 13.

Note: Returned trades are filtered according to the access rights of the querying user. This user should have access to both the proposing and responding agents in order to see a given trade.

Table 13: Trade Query Parameters

Input Information	Comments
Date Window Filter	Defines the date window into which the trade should fall.
Trade Status Filter	Allows you to define a filter of one or several trade statuses.
Agent Filter	A list of agents. This filter defines trades that have the specified agents as participants of those trades. This filter can be used to prepare an *Agent Trade Report.*

Table 13: Trade Query Parameters (Continued)

Input Information	Comments	
Sort Order	Sort Order and Comments	
	Proposal Agent	Trades are sorted according to the proposing agent.
	Response Agent	Trades are sorted according to the responding agent.
	Status	Sorted by trade `Status.`
	Proposal Paid Hours	Sorted by the number of paid hours in the proposal schedules.
	Response Paid Hours	Sorted by the number of paid hours in the response schedules.
	Submit Time	Sorted by the time of submission.
	Expire Time	Sorted by the response expiration time.
	Approve Time	Sorted by the time of approval.

Chapter

8 Performance

This chapter explains the Performance service, represented by the `WFMPerformanceService7<xx>Soap` interfaces, which retrieve performance data.

This chapter includes the following sections:

Performance Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, "About the WFM Integration API," on page 15 and Chapter 2, "Locator, Session, and Util," on page 23.

Performance Essentials

The Performance service is presented through the `WFMPerformanceService7<xx>Soap` interfaces. Use this service to retrieve performance information.

The main things you can do with the Performance service are as follows:

- Retrieve read-only contact-center performance data.
- Set the time zone in which you want to view the data.

Performance data includes actual contact-center performance, anticipated performance, and planned performance.

Actual Performance Data

The Performance service extracts actual performance information from the WFM database for each timestep. It retrieves the following information:

- Actual Interaction Volume
- Actual Average Handling Time
- Actual Service Factor
- Actual Abandoned Interactions Percentage
- Actual Working Agents
- Agent Difference
- Actual Average Speed Of Answer
- Actual Distributed Interactions
- Actual Handled Interactions

You can request information can be requested for the following levels:

- Activity
- Multi-site Activity (an aggregation of data for all associated activities)
- Site
- Business Unit
- Enterprise

Because performance information is stored for activities, when you select a different value, the value is calculated by aggregation.

Anticipated Performance Data

Since there are no values for actual performance information in the future, the Performance service provides anticipated values for all data types except Working Agents. Anticipated values are calculated based on previous actual and forecast values.

Planned Performance Data

The Performance service extracts forecast and scheduled information from the WFM database for each timestep for the activities you request. Forecast information is published to the database by the Forecast service. The Scheduler published scheduled information to the WFM database. The Performance service interprets this data as planned performance information and uses it for internal calculation of anticipated values. You can also request this data. You can request information about the following:

- Forecast Interactions Volume
- Forecast Average Handling Time
- Scheduled Service Factor
- Scheduled Abandoned Interactions Percentage

- Scheduled Average Speed Of Answer
- Scheduled Maximum Occupancy Percentage
- Scheduled Agents
- Coverage (Suitability)
- Required Service Factor
- Required Abandoned Interactions Percentage
- Required Average Speed Of Answer
- Required Agents
- Required Agents Difference
- Calculated Agents
- Calculated Agents Difference
- Under Required Staffing
- Under Calculated Staffing
- Over Required Staffing
- Over Calculated Staffing

Note: For explanations of these metrics, see the Performance section of *Workforce Management 7.5 Web for Supervisors Help*. You can open the Help file from any WFM Web for Supervisors window or access a stand-alone version on the Genesys Technical Support web site.

You can request information can be requested for the following levels:

- Activity
- Multi-site Activity (an aggregation of data for all associated activities)
- Site
- Business Unit
- Enterprise

Because performance information is stored for activities, when you select a different value, the value is calculated by aggregation.

Forecast Objectives and Parameters

The Performance service uses objectives and parameters to calculate some anticipated, scheduled, and required values. Service objectives and parameters are used to build forecast data (*requested* service objectives). The Performance service also calculates the actual service objectives achieved after the forecast data was built. Requested and actual service objectives normally differ because WFM tries to satisfy all the specified objectives to achieve better-than-requested performance. Objectives and parameters are set for each activity for every timestep (or, in some cases, for a specified period of time).

Retrieving Performance Data

Use the getPerformanceData method to retrieve actual and planned contact-center performance data—that is, data about how your contact center is performing based on the service objectives used in the forecast.

The PerfTest.java code example shows how to retrieve this data. See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Chapter

9 Adherence

This chapter explains the Adherence service, represented by the `WFMAdherenceService7<xx>Soap` interfaces, which retrieve agent-adherence data.

This chapter includes the following sections:

Adherence Prerequisites

To follow the discussion in this chapter, you will need the *Workforce Management 7.5 Integration API Reference* (located on your product CD and the Developer Documentation CD).

The discussion in this chapter also assumes that you have read Chapter 1, “About the WFM Integration API,” on page 15, and Chapter 2, “Locator, Session, and Util,” on page 23.

Adherence Essentials

The Adherence service is presented through the `WFMAdherenceService7<xx>Soap` interfaces. Use this service to retrieve agent-adherence data.

The main things you can do with the Adherence service are as follows:

- Retrieve agent-state data.
- Retrieve all adherence events. Adherence events are occasions when an agent’s actual state is compared with the scheduled state. Agents are compliant, nonadherent, or severely nonadherent, depending on whether they are out of sync with their scheduled state, and, if so, for how long.

Genesys Agent States

At any given moment every agent is at some state assigned by Genesys Framework. See Table 14 for the predefined set of available agent states. Also, the agent state may have user-defined reason (aux) code attached, to further specify what the agent is actually doing.

The *Workforce Management 7.5 Configuration Utility Help* discusses Genesys agent states and how they are used in WFM. It also explains how WFM groups schedule states in Schedule State Groups.

Note: Every agent's state is tracked in real-time and is logged to the WFM database by WFM Data Aggregator. The database format is proprietary. You cannot access real-time agent state data, although historical agent-state information is available through the Adherence service.

The list of predefined Genesys agent states can be obtained programmatically from the Configuration service by retrieving the Agent State object. Agent states are as follows:

Table 14: Agent States

State ID	State
0	NotMonitored
1	Monitored
2	LoggedIn
3	OnHook
4	WaitForNextCall
5	OffHook
6	CallDialing
7	CallRinging
8	NotReadyForNextCall
9	AfterCallWork
10	OfflineWorkType2
11	BreakType1
12	BreakType2

Table 14: Agent States (Continued)

State ID	State
13	CallOnHold
18	CallUnknown
19	CallConsult
20	CallInternal
21	CallOutbound
22	CallInbound
23	LoggedOut

Adherence Agent State Record

The historical agent state log consists of the following information that, together, is called the Adherence Agent State.

Table 15 shows the parameters used to retrieve adherence events from the Adherence service.

Table 15: Agent State Record

Field	Description
Agent ID	The Config Service is used to obtain detailed agent information given an Agent ID. See Agent Object for description.
State ID	Genesys Agent State ID
Reason	User-defined reason code(s) attached to the state.
Start Time	State start date/time
End Time	State end date/time

Adherence Events

Historical agent-state information can be compared against an agent’s published schedule to check the agent's adherence to the schedule. Using the WFM Configuration Utility, you can configure limits to the amount of time an agent can be out of sync with the schedule before he or she is considered nonadherent.

A period of time during which the agent is either continually adherent or continually nonadherent is called an *Adherence Event*. The Adherence service provides an interface for accessing dynamically-generated adherence event records for the time period you specify.

Table 16 shows the parameters used to retrieve adherence events from the Adherence service.

Table 16: Adherence Event Record

Field	Description
Agent ID	The Config Service is used to obtain detailed agent information given an Agent ID. See Agent Object for description.
Start Time	State start date/time
End Time	State end date/time
Adherence Agent State Record(s)	Describes Agent State for the duration of the Adherence Event
Scheduled Agent State Record(s)	Describes Scheduled Agent State for the duration of the Adherence Event. See description of Schedule Service for more details.
Status	Tells whether an event was adherent or non-adherent to the schedule.

Retrieving Adherence Events for a Day

In most cases, agent-adherence events are presented grouped by calendar days in reports. The Adherence service also provides agent-adherence events, grouped by day, as a separate record.

Table 17 shows the parameters used to retrieve adherence events for a day from the Adherence service.

Table 17: Adherence Day Event Record

Field	Description
Agent ID	The Config Service is used to obtain detailed agent information given an Agent ID. See Agent Object for description.
Date	Calendar date designating a day

Table 17: Adherence Day Event Record (Continued)

Field	Description
Adherence Event Records	A list of Agent Adherence Events that have occurred within this day.
Non-Adherence Duration	Total of the non-adherence state duration for the day (in seconds).
Out of Schedule Non-Adherence Duration	Total out of schedule non-adherence state duration for the day (in seconds).
Schedule Duration	Total schedule duration for the day (in seconds).
Adherence Percentage	A percentage of adherent state time during the day.

Retrieving Agent State and Adherence Event Data

Agent-state data shows what the agents' actual states were, as opposed to the schedule states. For example, if an agent is scheduled for a break at 2:00, but leaves for the break at 1:56, the agent's actual state differs from the schedule state. Therefore, he or she is nonadherent for four minutes.

The `AdhTest.java` code example shows how to retrieve agent-state data. It also shows how to retrieve information on all compliant events, nonadherent events, and all severely nonadherent events for the specified place and time.

See "Using the Code Samples" on page 19 for the location of the code samples and how to use them.

Note: The code samples are intended for use as guidelines for building your own application. They are as error-free as possible, but they are not guaranteed or supported by Genesys. These examples cannot be joined together to create a working application.

Index