Agent Interaction SDK Java Developer Guide

About Agent Interaction (Java API)

12/12/2025

Contents

About Agent Interaction (Java API)

This chapter introduces the Agent Interaction Java API, its components, features, and scope of use.

Library Overview

The Agent Interaction (Java API) lets you build applications to control and manage voice and multimedia interactions issued by, or intended for, a contact center agent.
The 7.6 release is backward-compatible with the 7.x releases. It is backward-compatible with the 6.5.6 release for voice features if your application uses voice features only.

Components

The Agent Interaction (Java API) comprises the following:

- The Agent Interaction Layer (AIL) library, highly portable, is written entirely in the Java language, delivered as a set of `.jar` files.
- The *Agent Interaction SDK 7.6 Java API Reference*, which is an HTML tree in the `docs/` directory of the installed product directory.
- The Agent Interaction SDK Java Examples, which is a set of code examples that exercise some important features of the API, delivered in .zip and .tar.gz format.
- A set of Application Blocks available on the Product CD. See the *Agent Interaction SDK 7.6 Application Blocks Guide* for further details.

AIL Library

The heart of the Agent Interaction (Java API) is the Agent Interaction Layer (AIL) library. The library can be thought of as two parts: a library core and an API.

Library Core

The library core manages connections to Genesys solution components. It is designed to work only with the set of telephone system switches that are described in the *Interaction SDK 7.6 Java Deployment Guide*.
The core maintains connections to components of the Genesys Framework, Genesys Multimedia, Outbound Contact Solution 7.*x*, and other Genesys solutions.
The core internally maintains objects used by your applications. These internal objects are not directly available through the API. Rather, the library maintains an *AilFactory* object, which provides you with access to API objects.

> ### Warning
> You cannot directly use or modify classes and methods in the library core. The Agent Interaction (Java API) is restricted to the use of the library API as described in the *Agent Interaction SDK 7.6 Java API Reference* delivered with this product. See for downloading API references.

Library API

The library API provides access to the features, statuses, and events managed by the library core. It provides a complete set of classes and interfaces to handle data of the underlying Genesys solutions with which you can design your own multimedia application.
The example Java programs that accompany this product illustrate the use of the interfaces for the most commonly used objects and their events (DNs, places, agents, contacts, and so on), as well as interactions for such services as voice and callback.

Scope of Use

The Agent Interaction (Java API) enables you to develop applications for the following purposes:

- Creating a contact-center agent desktop application for Genesys software implementations.
- Integrating Genesys software with third-party software.
- Creating other, specialized applications specific to your needs.

Typical usage scenarios include:

- Managing agent login activity.
- Monitoring agent status.
- Handling e-mail and collaborative e-mail interactions: sending, receiving, replying.
- Handling voice interactions: calling, receiving, callback, outbound.
- Handling chat interactions.
- Handling open media interactions.
- Handling outbound campaign participation.
- Handling collaboration sessions.
- Accessing the Standard Response Library.

Your application can handle any inbound interaction, regardless of media:

- Answer a phone call.
- Accept an e-mail.
- Accept a request for a chat session.
- Accept an open media interaction.

Your application can initiate outbound interactions, regardless of media:

- Make a phone call.
- Send an e-mail.
- Make a preview outbound call.
- Make a predictive outbound call.
- Submit an open media interaction.

The AIL library offers you two primary modes of deployment or application development:

- A stand-alone application client. Your application code binds with the AIL library at runtime.
- A server application in *n*-tier architecture. You can write a server application that binds with the AIL library at runtime, or you can design your application to work within a container, such as Tomcat, to respond to web-browser client requests.

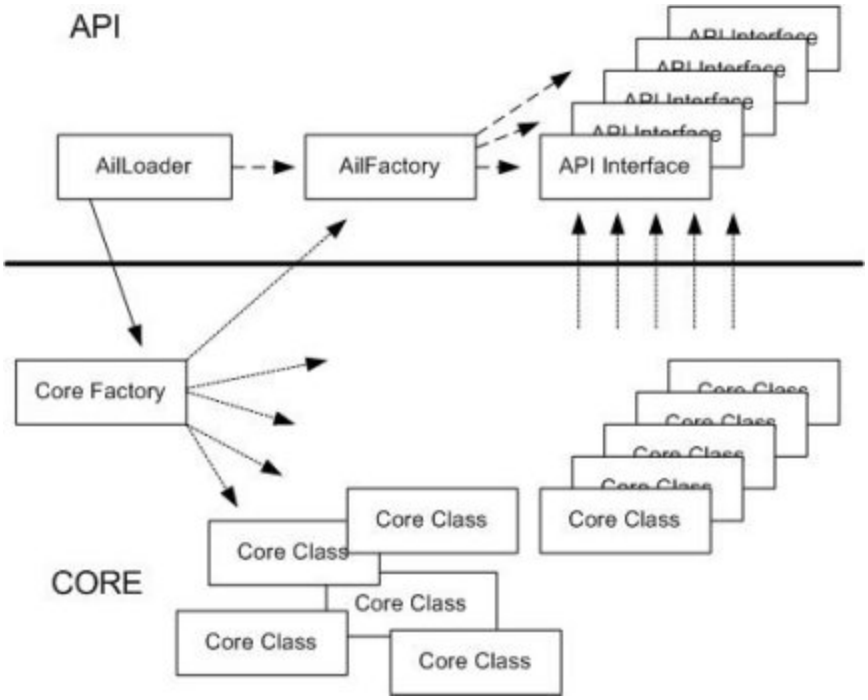
Refer to the *Interaction SDK 7.6 Deployment Guide* for configuration information pertinent to these two modes.

Architecture

The AIL library core is responsible for maintaining connections to servers, maintaining context, managing media, consolidating data, obtaining real-time object information, and providing switch facilities.
The API exposes objects, such as DNs, interactions, and agents, as interfaces that give access to all necessary information. The core manages the objects with respect to state machines that guarantee that the model is coherent with other Genesys components (for example, multimedia solutions) across supported switches. Changes in the object states are available through events.

Interfaces to Core Objects

You do not access core objects of the Agent Interaction (Java API) library directly. Rather, you get interfaces on them using the `AilFactory`.
Your application uses the `AilLoader` class to get an interface to the internal `AilFactory` object. The `AilLoader` class' methods establish connections at application startup, and provide access to the `AilFactory` .
The internal `AilFactory` object is itself available as an interface, through which you access the core Agent Interaction Layer factory object. The `AilFactory` object is a singleton.
Because of its singleton design, only one instance of the core `AilFactory` object exists at runtime. All `AilFactory` interfaces obtained through the `AilLoader` refer to this same object.
The `AilFactory` instantiates internal classes and makes them available through interfaces, as illustrated below. You do not instantiate objects directly by using a `new` instruction. You rather get an interface by calling a `get()` method.



Core Factory Interface

The library provides a reference system of unique object IDs that are standard `String` objects. Thus, your application manipulates each object by passing its unique ID as a parameter in methods of the `AilFactory` interface.

Core Features

The library core is designed around the following features:

- Switch Facilities
- Multithread Implementation
- Synchronization and Error Handling
- Persistence
- Cache Mechanisms
- Back-End Server Connections Management (hot standby, ADDP)

Switch Facilities

The Agent Interaction (Java API) provides a general state model for all media, including voice, which relies on T-Server–switch pairings. Therefore, the library core takes into account switch-specific features on suitable switch facilities. For extensive details about writing applications with respect to switch features, see Switch Facilities.

Multithreaded

The Agent Interaction Layer library is thread-safe and can therefore be run in a multithreaded environment.
On startup, it creates a thread that maintains all Genesys server connections, a thread for processing T-Server requests, a thread for publishing events to the API, and so on. Events from a given T-Server are always forwarded to the API listeners in the same order as they were received from the T-Server.

Synchronization and Error Handling

Usually, communication with the underlying servers is asynchronous. So an acknowledgment or result of a request is returned through the common flow of events sent by a server.
To relieve the developer from managing such asynchronous requests, methods of the API are made synchronous: the server reply is awaited before the requesting method returns.
The API uses exceptions for notification about standard errors, communication errors with the Genesys servers, unavailable actions errors, or status errors.
Moreover, final results of actions are forwarded as events to the registered listeners.
Because the API is synchronized, a request will block the current thread until it returns. For voice applications, the request will block until the request has been completed on the switch or an error occurs.
A request that goes up to the switch or the database might take time. Because this is not what you want in a single-threaded application, you may want to develop a multi-threaded application.
The library implements a timing loop that you can configure in the constructor parameters at the time you create a new `AilLoader` . This is a defense mechanism to allow application recovery following switch, network, T-Server, or other connection failures.
In the event of a failure, an exception is thrown. See the Javadoc API Reference for details.

Persistence

The Agent Interaction Layer library can establish a link with a Contact Server and provides a high-level representation of its objects through the interfaces of the API.
Note that when you have a reference on an object, it can be modified by another thread. For example, a T-Server event can change the status of an interaction. If you call the `getStatus()` method twice on an `InteractionVoice`, the other party might have released the call in between. Then the first `getStatus()` method will return `TALKING` and the second `getStatus()` method will return `IDLE` .
The objects `Contact` and `Interaction` have a representation in the Contact Server and thus can be saved. They implement the `Savable` interface that controls the coherence between the objects and the Contact Server.
These objects can be manipulated with their identifiers.
For a client application to directly retrieve an `Interaction` object from the Contact Server, it can use the `getContactServerId()` method to get the internal DBID of the `Interaction` object.

Cache Mechanism

To improve performance one step further, two different cache mechanisms are implemented:

- An interaction cache on current manipulated interactions.
- A configuration cache for the objects found in the Configuration Layer, such as objects implementing the interfaces `Agent` , `Dn` , or `Place` .

Depending on the way you configure your environment, the configuration cache can be entirely preloaded on startup. This creates a load time proportional to the amount of data in the Configuration Layer.
The configuration cache is fully dynamic: a modification in the Configuration Layer is immediately updated in the cache.

Connectivity and Internal Features

The library core provides the event mechanism, through which your Agent Interaction (Java API) application can notify users about servers' statuses (notably, the loss of a connection).
The library core can maintain connections to multiple T-Servers.
The library core is designed to work in a single-tenant environment. It is possible to create applications that work in multi-tenant environments, but in this case, Configuration Layer objects that your application uses must be specified in the application's Tenants tab (in Configuration Manager), and these names must be unique. See the *Interaction SDK 7.6 Java Deployment Guide* for details.

Framework Compatibility

The Agent Interaction (Java API) connects to the following Genesys servers within the Genesys Framework:

- Configuration Layer—The Configuration Layer stores configuration information (such as application parameters) and objects' descriptions (such as DNs, places, and persons). The library core monitors the Configuration Layer to respond to modifications. The library provides full integration with Genesys Configuration Layer objects such as Agent, Place, and DN.
- Stat Server—this core component keeps track of resource state for your Genesys environment.

> **Important**
> Since 7.6.4, AIL cannot connect to more than one Stat Server.

- T-Server—The Telephony Server handles telephone requests and events by communicating with switches.

For voice-only mode, your application should connect to the Configuration Layer and to at least one T-Server. For details about supported switches, refer to the *Genesys Supported Media Interfaces* document.

PSDK Application Template Application Block

In 7.6.6, Agent Interaction SDK integrates the PSDK Application Template Application Block and provides the following additional features:

- TLS Support for connections to Genesys Servers.
- Policy International Data Support
- Management of AIL dependencies on JDK pluggable service providers.

For further details, read the PSDK Application Template Application Block documentation.

If you wish to replace the default XML parser with the XML JVM options, you will need the following set of JVM options:

```
-Dcom.genesyslab.platform.commons.xml-doc-builder-
factory=com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl
-Dcom.genesyslab.platform.commons.xml-transformer-
factory=com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl
-Dcom.genesyslab.platform.commons.xml-xpath-
factory=com.sun.org.apache.xpath.internal.jaxp.XPathFactoryImpl
```

AIL also takes those options into account for its internal needs and always uses the SUN XML parser.

Multimedia Compatibility

The Agent Interaction (Java API) is compatible with Genesys Multimedia, and provides full multimedia support for voice, e-mail, chat, and open media interactions.
The Agent Interaction Layer library connects to the following Genesys Multimedia servers:

- Interaction Server. This server manages non-voice interaction information.
- Chat Server. This server manages chat interactions between agents and web visitors.
- Universal Contact Server (UCS). This database server is used to retrieve and store e-mails, history, and contact information. You can manipulate the contact history and the standard response library.

For e-mail handling, your application should connect to a Configuration Layer, and a Universal Contact Server and an Interaction Server (both included with Multimedia).
For chat handling, your application should connect to a Configuration Layer, a Chat Server, an Interaction Server, and a Universal Contact Server (all three both included with Multimedia).
For open media handling, your application should connect to a Configuration Layer, an Interaction Server and optionally a Universal Contact Server (both included with Multimedia).

Outbound Campaign Support

The Agent Interaction (Java API) connects to the Genesys Outbound Solution through the Outbound Campaign Server. This server controls and organizes outbound campaigns.
For outbound campaign handling, your application should connect to, if using voice outbound, to a Configuration Layer, an Outbound Contact Server, a T-Server, and, optionally, UCS. If you are using outbound proactively, you should connect to Interaction Server and at least one T-Server.

Voice Callback Support

The Agent Interaction (Java API) connects to the Genesys Universal Callback Solution through the Callback Server. This server controls and organizes callback records.
For Voice Callback handling, the Agent Interaction (Java API) should connect to a Configuration Layer, a Callback Server, and at least one T-Server.

Multi-Tenancy Support

The Interaction SDKs are not suited for multi-tenant deployments. Although you can use them for a given tenant in a multi-tenant environment, you would need a separate instance of your application

for each tenant using it. (As an alternative, the Genesys Platform SDK supports multi-tenancy.)