Agent Interaction SDK Java Developer Guide

PSDK Bridges

12/14/2025

Contents

PSDK Bridges

The Voice and Configuration PSDK Bridges are two distinct APIs that are integrated into the to the Agent Interaction Java API. They are intended to be used for implementing additional controls on objects that the Agent Interaction Java API does not handle.
Using this API requires an advanced level of coding experience with Agent Interaction Java API. Any custom code on top of these bridges requires requires testing in a production-like environment prior to any deployment.

Guidelines

As mentioned above, the PSDK bridges are particular APIs integrated into the Agent Interaction Java API. They provide access points to PSDK protocol instances in order to implement additional controls through the Platform SDK API:

- `com.genesyslab.platform.voice.protocol.TServerProtocol`
- `com.genesyslab.platform.configuration.protocol.ConfServerProtocol`

As explained in the following sections, the use of these classes is restricted to the Call Center objects that do not interfere with the AIL Core. For more details about these protocol classes, refer to the *Platform SDK 7.6* documentation.

Protocol Instances

When your application calls the `PsdkConfigBridge.getConfigServerProtocol()` or the`PsdkVoiceBridge.getTServer()` method, it retrieves the PSDK protocol instance that the AIL Core uses to communicate with the Configuration Server or with the specified switch.
Even when using the bridge, AIL manages connections, disconnections, HA and ADDP. (In any case, as a general guideline, your application should not deal with any of them through the AIL's PSDK bridges.)

> **Warning**
> Do not call the `close()` methods of the protocol instances that you get from the PSDK bridges. You would close the AIL's connections and make unavailable the server and the associated features until AIL reconnects.

Use special care when working with the Platform SDK since it exposes low-level APIs for the underlying servers. Because you deal directly with the PSDK protocol instances and not with some interfaces as for other Call Center objects (Agent, Place, DN, and so on), the AIL Core is not aware of requests that you make through PSDK bridges.

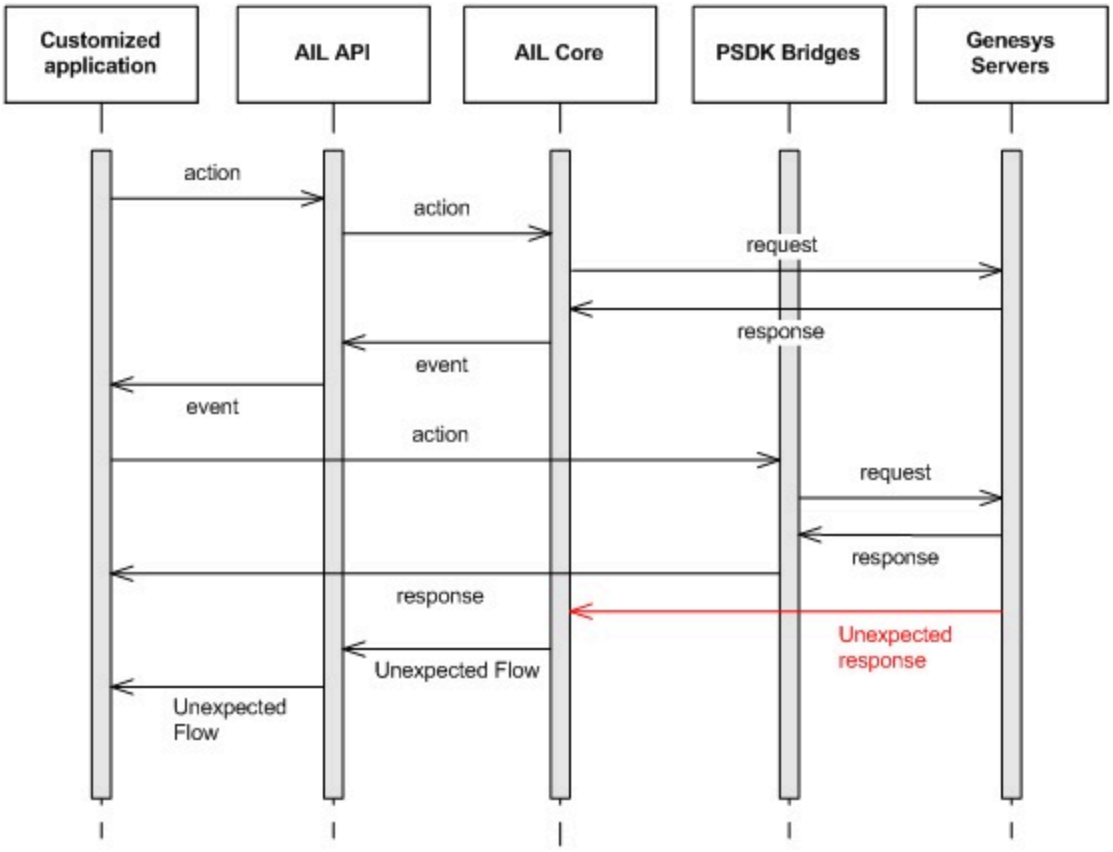
The PSDK Config Bridge' usage is restricted to monitoring objects or retrieving information. The PSDK Voice Bridge, in addition to monitoring features, supports some voice requests. The customization of

your application with the PSDK bridges can bring some overhead with the event flow or put AIL out of synchronization, as explained in the following sections.

> **Important**
>
> Prior to the deployment of this customization, you must perform unit tests on your application, and also load tests in production-like environments.

Message Flow

When you deal with the PSDK bridges, you create a new message flow using the `com.genesyslab.platform.voice.protocol.TServerProtocol` and `com.genesyslab.platform.configuration.protocol.ConfServerProtocol` classes.
The PSDK bridges' protocol instances enable you to send requests and, for those requests, if you get responses, AIL receives those too. As a result, AIL generates an event flow according to the responses received, as it does when no PSDK bridge is implemented. The difference is that you receive responses low-level requests that you might not otherwise. Usually AIL only handles messages that are responses to its own requests.
For this reason, AIL may generate unexpected events from an agent application's point of view. It is not possible to assume what the PSDK's additional message flow is, because it strictly depends on how your application makes use of the PSDK bridges.



Message Flow

PSDK Config Bridge

- **Description:** The PSDK Config Bridge is an access point to the Configuration Platform SDK which enables you to monitor objects in the Configuration Layer of your Genesys environment. Use special care when working with the Platform SDK since it exposes low-level APIs for the underlying servers.

For instance, you should avoid sending modification requests to the Configuration Server because the AIL Core would not be notified of these changes, and the AIL cache would be out of synchronization.

PSDK Voice Bridge

The PSDK Voice Bridge is an access point to the Voice Platform SDK which enables you to monitor voice interactions from a traditional or IP-based telephony device. In addition, you can also perform requests with the following packages:

- `com.genesyslab.platform.voice.protocol.tserver.requests.dtmf`
- `com.genesyslab.platform.voice.protocol.tserver.requests.special`
- `com.genesyslab.platform.voice.protocol.tserver.requests.queries`

- `com.genesyslab.platform.voice.protocol.tserver.requests.iscc`
- `com.genesyslab.platform.voice.protocol.tserver.requests.voicemail`

Use special care when working with the Platform SDK since it exposes low-level APIs for the underlying servers. Although AIL handles unsolicited telephony events, modifying devices or interactions may lead to data inconsistency, for instance, in connection with the result of the`isPossible()` method calls or in reference IDs that AIL creates for `Interaction` instances.

> **Warning**
> Do not modify DN registration with the PSDK Voice Bridge.

Managing PSDK Listeners

Even when using the bridge, AIL manages connections, disconnections, HA and ADDP. (In any case, as a general guideline, your application should not deal with any of them through the AIL's PSDK bridges.)

> **Warning**
> Do not call the `close()` methods of the protocol instances available through the PSDK bridges. That would close the AIL connection and make unavailable the server and the associated features till AIL reconnects.

In case of disconnections, the listeners (that your application registered to get protocol messages) are removed. So, your application must listen to configuration and telephony service statuses to get notified of disconnections. When AIL reconnects, the customer application must register its listeners again to get PSDK protocol messages. See also the section Steps for Integrating PSDK Config Bridge.

Steps for Integrating PSDK Config Bridge

This section offers a short implementation example to handle properly the notification of protocol messages for the PSDK Config Bridge.

1. Create a new class implementing the ServiceListener interface.

```
public class ConfigBridgeAdapter implements ServiceListener {
 AilFactory myFactory;
```

In the constructor, add your listener to the listener list of the `CONFIG` service which monitors the connection to the Configuration Server, as shown below.

```
 public ConfigBridgeAdapter(AilFactory _myFactory) {
    myFactory = _myFactory ;
    myFactory.addServiceListener(ServiceStatus.Type.CONFIG,this);
```

```
}
```

2. Then, implement the handler for the status changed events. When the connection to the Configuration Server is in ON status, you must register a PsdkConfigBridge.Listener instance to get protocol messages. To correctly process events, you should use two class variables, including a QueuedExecutor instance:

```
// Class variables
QueuedExecutor mQueue = new QueuedExecutor();
PsdkConfigBridge.Listener mListener;

public void serviceStatusChanged( ServiceStatus.Type service_type, java.lang.String
service_name, ServiceStatus.Status service_status) {

        // handle Service.Status on disconnections
        if(service_status.toInt() == ServiceStatus.Status.OFF_) {
                //...
                // TODO: implement what your application should do // when connection is
turned OFF
                //...
        } else if(service_status.toInt() == ServiceStatus.Status.ON_)
        {
                registerBridge();
        }
}

public void registerBridge()
{
        // Create the listener
        mListener = new PsdkConfigBridge.Listener() {
                public void handle( Message message ) {
                        mQueue.execute( new Runnable() {
                                public void run() {
                                        myHandler( message );
                                }} );
                }};

                PsdkConfigBridge.addConfigServerListener( mListener );
                // Your application now receives protocol messages
}

public void myHandler( Message message ) {
        /**
         * TODO Implement the message handling
         **/
}
} // End of ConfigBridgeAdapter class
```

As soon as you add your ConfigBridgeAdapter instance to the listener list of the CONFIG service, your class is notified of the connection status and is added to the protocol's listener list.

3. Now, you can retrieve the protocol instance and start customizing your application.

```
// Get the factory
AilFactory my_factory = AilFactory.getAilFactory();
// Listen to CONFIG service
ConfigBridgeAdapter myAdapter = new ConfigBridgeAdapter(my_factory);

// Get Protocol instance
com.genesyslab.platform.configuration.protocol.ConfServerProtocol
ailConfigProtocol = PsdkConfigBridge.getConfigServerProtocol() ;
```

```
    /**
    * TODO: implement customization
    */
```