

# Agent Interaction SDK Java Developer Guide

Expert Contact

12/14/2025

# Contents

# Expert Contact

Handling expert contact information does not require handling a particular interaction type. Your application manages voice interactions. Implementing expert contact is a matter of handling additional expert contact information that the Agent Interaction (Java API) provides to interactions. This chapter shows you how to deal with the expert contact service.

## Expert Contact Design

The Agent Interaction Java API supports features for an application that enables expert users, who are not part of an enterprise's Contact Center, to provide their expertise to Contact Center agents or customers.

### Usage Scenario

Contact center agents, in the course of responding to customers, sometimes need information beyond their training. In such cases, they can benefit from contacting people with special expertise. But typically, such experts (knowledge workers) are not part of the Contact center CTI infrastructure: their telephones connect to a switch that does not have a T-Server or Framework support. In such cases, their interactions with Contact center agents (or their customers) are not tracked, and neither the agents nor the experts can benefit from the information provided by Genesys Solutions.
In a site without a CTI link, the expert receives phone calls directly from a public network without the involvement of any Genesys platform components. Therefore such calls are not automatically monitored or controlled. For example, there is no way to detect the state of the expert's telephone (`Ready` , `OnCall` , and so on).
Genesys Expert Contact addresses this problem.

### Expert Contact Components

There are two major components involved in making expert contact work:

- A Genesys CTI-less T-Server, which works without monitoring a switch. This component provides a connection to the expert's desktop application and to a T-Server in the contact center.
- An Expert Contact desktop application that can use AIL library features to connect to a CTI-less T-Server and monitor its events for expert contact interactions.

### CTI-Less T-Server

A CTI-less T-Server provides a virtual CTI environment to track the expert's telephone states, to send messages to other Genesys server components, to handle data for current interactions, and to coordinate voice and data delivery to the expert's desktop application.
The CTI-less T-Server must communicate with a T-Server within the contact center infrastructure. It receives events from this T-Server and sends its own events to applications.

## Expert Contact Application

An expert contact application provides a means for an expert to reflect his or her call state. It can also present the expert with information from the Genesys Framework.
When an expert receives a call transferred from a Genesys supported contact center, the Genesys platform components communicate with the CTI-less T-Server, which notifies the expert's desktop application of an incoming phone call. The application presents an indication to the expert, who can choose to accept or reject the phone call.
If the expert accepts the phone call, the desktop application can present available information, including the contact history if there is a connection to a Genesys Contact Server.
As the phone call progresses, the expert must use the application to view this progress. The application passes the expert's activity to the CTI-less T-Server, which in turn passes the data to the Contact Center Framework components.
The application uses AIL library features to process events from the CTI-less T-Server.

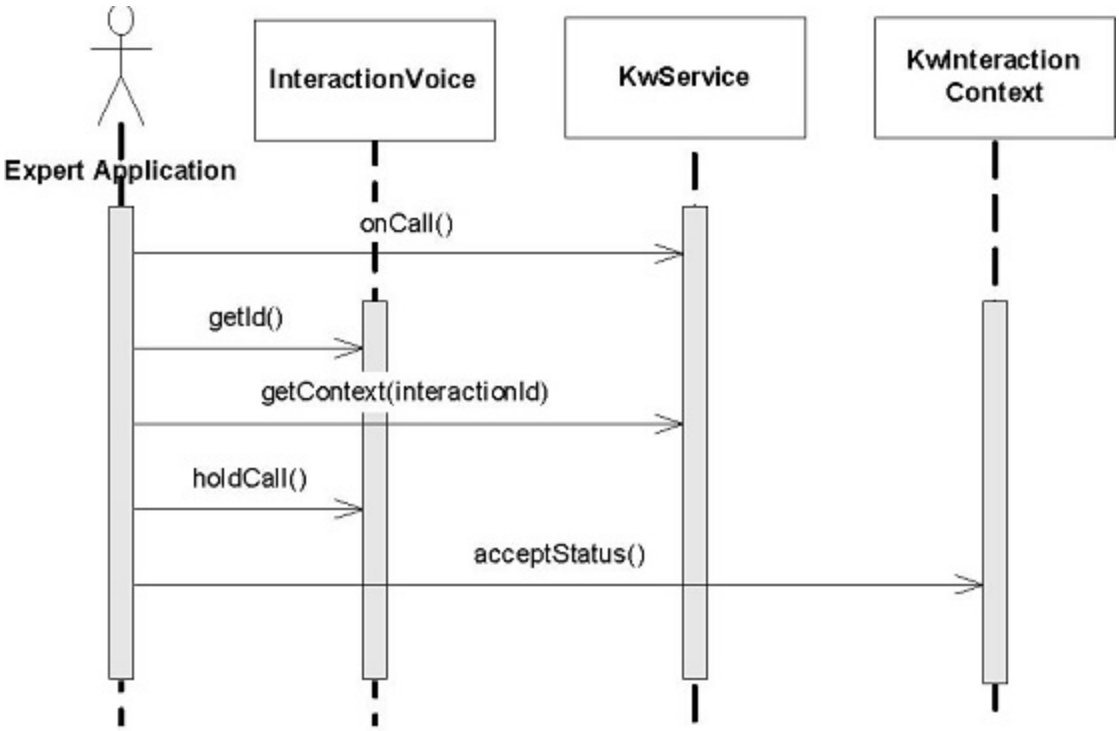
## Configuration

An expert contact desktop application built on the AIL library must connect to the CTI-less T-Server. It must also connect to a Configuration Layer that has information about the CTI-less T-Server, person information for the expert, and so on.
In addition to these configuration objects, the contact center site switch and T-Server must be configured for external routing.

> ### Important
> See the *Genesys Expert Contact 7.6 Deployment Guide* for instructions on how to configure a Genesys expert contact application in the Configuration Layer.

## Expert Contact Information

To get expert contact information, you deal with the `KwService` instance that you get by calling the `AilFactory.getKwService()` method. With `KwService` methods, you can get `KwInteractionContext` objects. Each `KwInteractionContext` object has an associated `InteractionVoice` on the CTI-less DN, and passes the expert's activity to the CTI-less T-Server, as shown below.




**Using Expert Features**

Your application uses `KwInteractionContext` methods to access the CTI-less T-Server, and uses `InteractionVoice` methods to emulate the expert's actions on the voice interaction.
For further information, refer to the *Genesys Expert Contact 7.x* documentation.

## On Call

The expert (or knowledge worker) can receive direct calls. The expert must notify the CTI-less T-Server of such calls. To perform that notification, your application must use the `KwService.onCall()` method.
The `onCall()` method sends a message to the CTI-less T-Server to send a new interaction to the specified DN. It also creates a `KwInteractionContext` instance for the voice interaction. Your application receives and manages this voice interaction as usual.
Then, the expert uses the voice interaction to report his or her actions on the call. For instance, when the expert manually answers the call, he or she uses your application to perform an `ANSWER` action on the voice interaction.

## Preview

The CTI-less T-Server can send a call to the expert. In this case, your application receives an `InteractionVoice` object associated with a `KwInteractionContext` object that has a `KwInteractionContext.Status.PREVIEW` status.
The expert can either accept or reject the call. Use the `accept()` or `reject()` method of the `KwInteractionContext` interface for this purpose.
A call to the `accept()` method sends the call to the expert's phone. Then, the expert uses the voice

interaction to report his or her actions on the call.

## Status Request

If your application has set listeners, your application can receive a `KwInteractionContextEvent` event propagating the `KwInteractionContext.Status.STATUS_REQUEST` of a `KwInteractionContext` object. This periodically happens when the CTI-less T-Server requests that the expert set up his or her voice interaction's status.
The expert can choose between the `KwInteractionContext.Action.CONFIRM_STATUS` or `KwInteractionContext.Action.REJECT_STATUS` actions.
A call to the `KwInteractionContext.confirmStatus()` method indicates that the expert is still on call. A call to the `KwInteractionContext.rejectStatus()` method indicates that the expert has terminated the call and the voice interaction is automatically released.

## Easy New Call and Auto Mark Done

When the expert makes a phone call, he or she must also create, and then dial, a voice interaction on his or her CTI-less DN using your application. This creates a `KwInteractionContext` instance for the voice interaction.

Your application processes the creation of this voice interaction as it would process the creation of a standard voice interaction. For further information, see Six Steps to an AIL Client Application.
Depending on your AIL configuration settings, your application can benefit from the `Easy New Call` feature. This feature changes the voice interaction's status to `TALKING` as of the interaction's creation on CTI-less DNs. The interaction's creation is therefore less time-consuming for the expert.

> ### Important
> To activate the `Easy New Call` feature, set the `easy-newcall` option to `true`. See the *Interaction SDK Java Deployment Guide* for further details.

When the expert hangs up a call, he or she should release the voice interaction and mark it as done.
Depending on your AIL configuration settings, your application can benefit from the `Auto Mark Done` feature. This feature automatically marks released interactions as done for CTI-less DNs.

> ### Important
> To activate the `Auto Mark Done` feature, set the `auto-markdone` option to `true`. See the *Interaction SDK Java Deployment Guide* for further details.

## Re-Route

Depending on your AIL configuration settings, the expert is able to re-route calls. See the *Interaction SDK 7.2 Java Deployment Guide* for further details.
If you properly set `kworker` routing options, your application can use the `KwInteractionContext.reroute()` method to notify the CTI-less T-Server of the voice interaction's

routing.

## Steps for Writing an Expert Contact Application

Now that you have been introduced to the expert contact feature's design, it is time to outline the steps you will need to work with its events and objects.
As specified in the previous section, expert contact data do not interfere with voice interaction management. Modifications in your voice application consist in a few adds-in to handle expert contact data and provide the user with expert contact features (for instance, by implementing a GUI panel dedicated to expert contact.)
There are five basic things you will need to do in your AIL applications:

- **Implement a KwInteractionContextListener listener** to get notified of changes in `KwInteractionContext` objects. Here is how a SimpleContextListener class would do this:

```
public class SimpleContextListener implements KwInteractionContextListener {
        //...
        public void handleKwInteractionContextEvent( KwInteractionContextEvent event)
        {
                KwInteractionContextEvent context = event.getInteraction();
                // update your application with expert context information
                // for instance, buttons for actions on the expert context
                //...
        }
```

- **Get a KwService interface** to access expert contact data. For instance, you could modify one standalone code example by declaring a private `kwService` variable, then by adding the following code snippet in the constructor method:

```
Class SimpleExpertContactExample implements PlaceListener
{
        KwService kwService;
        //...
        public SimpleExpertContactExample(AilFactory ailFactory)
        {
                kwService = ailFactory.getKwService();
                if(callbackService.isAvailable())
                {
                        //...
                }
        }
        //...
}
```

- **Set up button actions** (or actions on other GUI components) tied to expert contact features, according to the `KwInteractionContext` interface's methods.
- **Check if interactions own expert contact information** in the implemented `handleInteractionEvent()` methods. Create a thread that manages the interaction event and update your application with expert contact information.

```
Interaction interaction = event.getInteraction();

//Getting the associated expert context (if any)
KwInteractionContext kwInteractionContext = myKwService.getContext(interaction);
```

```
//Add a listener for changes in the expert context
kwInteractionContext. addKwInteractionContextListener(new SimpleContextListener()) ;

// update your application with the kwInteractionContext
//...
```