This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Agent Interaction SDK Java Developer Guide

Open Media Interactions

12/18/2025

Contents

Open Media Interactions

Open media interactions are multimedia interactions, that is, the `InteractionOpenMedia` interface inherits `InteractionMultimedia` interface. Other multimedia interactions are discussed in E-Mail Interactions and Chat Interactions.
This chapter presents `SimpleOpenMediaInteraction`, a new example that receives open media interaction.

Open Media Design

Contact center agents routinely work with applications that are separate from traditional CRM (Customer Relationship Management) or agent desktop applications. Genesys Open Media allows you to create custom media types that integrate this agent activity into the contact center workflow.
To use open media, you will need to define new interaction types in the Configuration Layer. After you do this, you can use the Media Interaction SDK to build client and server applications that manage them within the Genesys platform. Finally, you will use the Agent Interaction SDK to allow agents to process these interactions.
This section will provide an overview of the kinds of situations that lead application developers to use open media interactions. It will then outline some of the things you can do with open media using the Agent Interaction (Java API). Finally, it will give more in-depth descriptions of a couple of real-world open media use cases.

> **Important**
>
> For more information on the Media Interaction SDK, see *Media Interaction SDK 7.6 Java Developer's Guide*. For more information on multimedia interactions, see *Multimedia 7.6 User's Guide* and read the *Basic Interaction Models* chapter from the *Genesys Events and Models Reference Manual*.

Bridging the Contact Center and the Enterprise

There are many occasions when contact center agents could carry out work they do not normally do. Open media makes this a lot more productive than it would otherwise be.
For example, in the wake of a natural disaster, insurance companies tend to get high call volumes as people file their claims. At this point, you might want to have claims adjusters and others available to expand the contact center workforce.

But after most of the calls come in, the claims adjusters will have a lot of work to do on these new claims. With open media, you can route calls to adjusters when call volumes are high, and then you can route routine work from the adjusters to the contact center when call volumes are low.

Another common scenario is for contact center agents to handle faxed requests when they are idle. In this case, you could set up an interaction type that includes the fax data for the agent to process.

When the interaction is routed to the agent’s desktop, the agent can process the data as appropriate and mark the item as done.

There are several ways you could use open media in these situations. In the simplest cases, you might want to use open media interactions merely to route work to an agent. But in some cases, it might be even better to process information directly in the agent desktop, using the open media functionality of the Agent Interaction SDK. There will be an example of each of these scenarios later in this section, but first, here is a look at some of the things you can do with open media.

Basic Capabilities

Agent Interaction SDK makes it possible for you to perform actions like the following on open media interactions:

- Open an interaction (using an existing interaction as the parent interaction).
- Create a new interaction.
- Respond to a received interaction.
- Thread interactions.
- Forward an interaction.
- Transfer an interaction to another queue or place.
- Add an interaction to an existing history.
- Retrieve any open, pending, or terminated interaction—of any open media type—from a contact history database.
- Manage an interaction’s workflow.

> **Important**
>
> For more information on the open media–related functionality provided by the Agent Interaction SDK, see the Agent Interaction SDK API Reference’s entry on the InteractionOpenMedia class.

Routing Rejected Orders to an Agent

This example will show how you might use open media interactions simply as a routing mechanism, while having an agent continue to work in an application that is separate from his or her desktop. Agents for a telephone company might use many applications, including an order management application. When a customer calls in to order DSL (a form of high-speed Internet service), an agent must enter the customer information and submit the order. If the agent makes a mistake on the order form, the system will reject the order and send it back to the agent for correction.

Because the order management system is not linked to the contact routing platform, the incorrect order will sit at the agent's desktop—perhaps for hours—until there is a lull in inbound activity that

will allow the agent to address the problem.

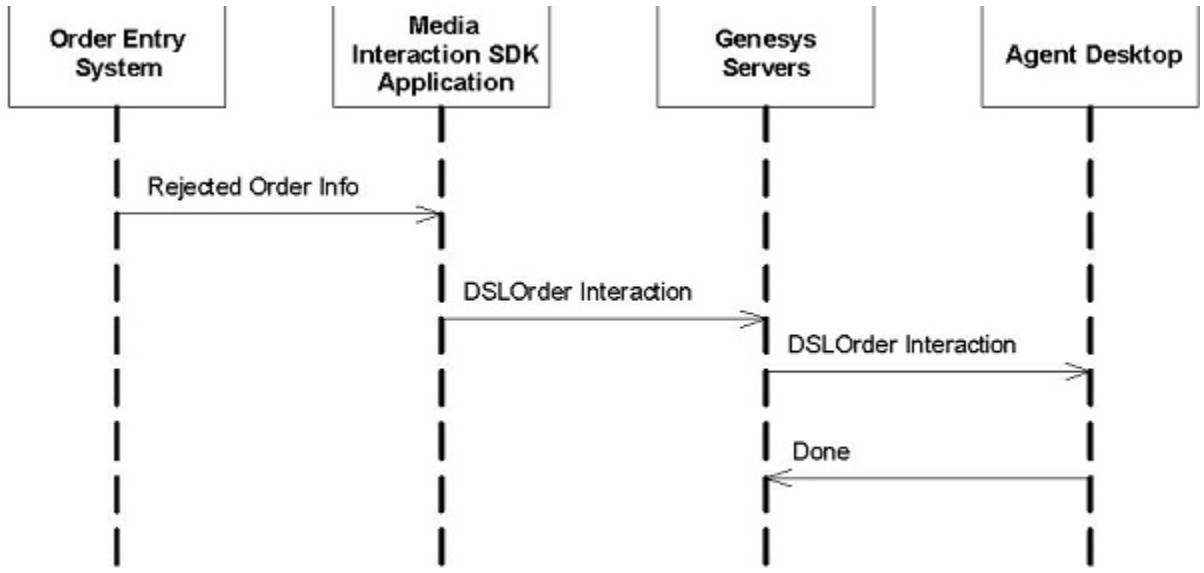
With the addition of Genesys Open Media, the scenario is different.

If the agent makes a mistake on an order, the rejected order can be submitted to the Genesys platform as a new activity to be queued and sent back to the agent. Depending on the priority that you set, the agent may be taken off the phone queue immediately and routed back to the order to make corrections.

Here is how you could use Genesys Open Media to create this kind of solution:

1. Define a new interaction type, DSLOrder, in the Configuration layer.
2. Set up the appropriate routing for the new interaction type.
3. Use the Media Interaction SDK to write an application that can receive information about rejected orders from the order management system.
4. Use the Agent Interaction SDK to write agent desktop functionality that allows the agent to mark the `DSL0rder` interaction as done. (Note that in this scenario, the agent is doing the order management work in the order management application. All that is required from the Agent Interaction SDK is the ability to mark the interaction as done.)

When an order is rejected, the order entry system will send information about the order to the new Media Interaction SDK application, which will create a new interaction of type `DSL0rder`. The interaction will be sent to the Genesys servers for processing and they will route the interaction to the appropriate agent, as shown below. When the agent has corrected the order, he or she will mark the interaction as done.



Alerting an Agent to Process a Rejected DSL Order

See SimpleOpenMediaInteraction for a code example that shows how to write an agent desktop application that can mark an open media interaction as done.

Working on a CRM Case

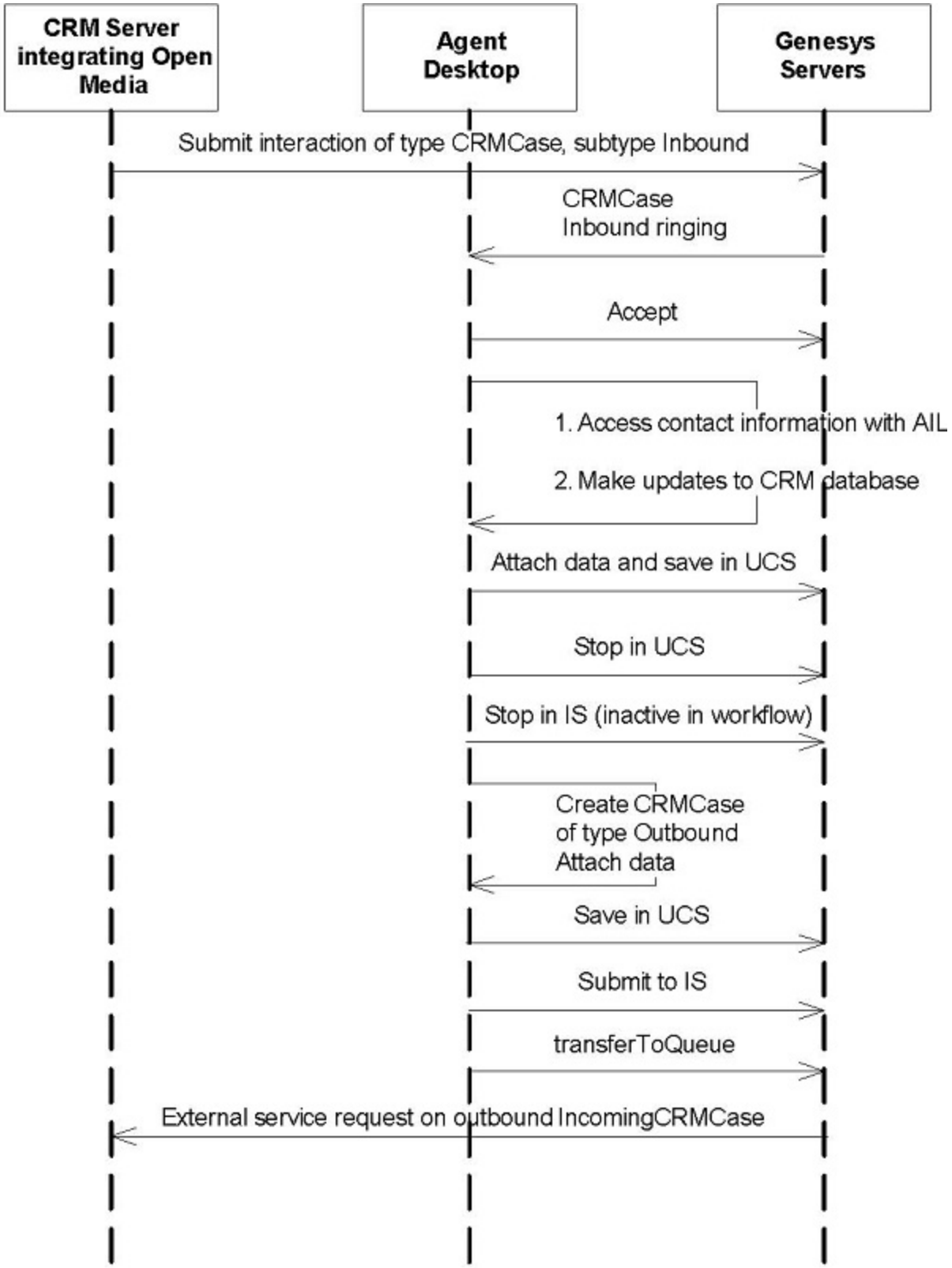
Sometimes it is not enough to use open media as a routing and notification mechanism. There are many cases where it makes more sense to write detailed interaction-handling functionality directly into the agent desktop.

For example, you can use open media interactions to allow contact center agents to handle incoming fax data for use by a Customer Relationship Management system. This could be done in a way that is very similar to the preceding example, but in this case, you might prefer that the interactions you create would also carry data for the agent to process right in his or her agent desktop application.

As in the example above, you would need to define a new interaction type, perhaps called `CRMCase`. You would also need to set up the appropriate routing and write an application that uses the Media Interaction SDK. This application would attach CRM data to the `CRMCase` interaction.

But the Agent Interaction SDK functionality you write would be a bit different, as it would do more than just mark the interaction `Done`. Here is the process the interaction would follow, as shown in Handling a CRM Case Using Open Media:

- The CRM server submits a `CRMCase` interaction of type `Inbound` to the Genesys servers.
- The interaction rings at the agent's desktop.
- The agent accepts the interaction.
- The agent accesses contact information from the Universal Contact server, using the Agent Interaction SDK.
- The agent desktop updates the CRM database.
- The agent desktop saves the interaction in the UCS database, using the Agent Interaction SDK.
- The agent desktop stops the inbound interaction in UCS.
- It also stops the interaction in Interaction Server.
- The agent desktop creates a reply to the inbound interaction. This reply is a `CRMCase` interaction of subtype `Outbound`.
- The agent desktop attaches data to the outbound `CRMCase` interaction, using data from the inbound transaction, as appropriate.
- The agent desktop submits the outbound interaction (to make it active in the Interaction Server workflow).
- The interaction is transferred to a queue.


CRM Server integrating Open Media
Agent Desktop
Genesys Servers
Submit interaction of type CRMCase, subtype Inbound
CRMCase Inbound ringing
Accept
1. Access contact information with AIL
2. Make updates to CRM database
Attach data and save in UCS
Stop in UCS
Stop in IS (inactive in workflow)
Create CRMCase of type Outbound Attach data
Save in UCS
Submit to IS
transferToQueue
External service request on outbound IncomingCRMCase

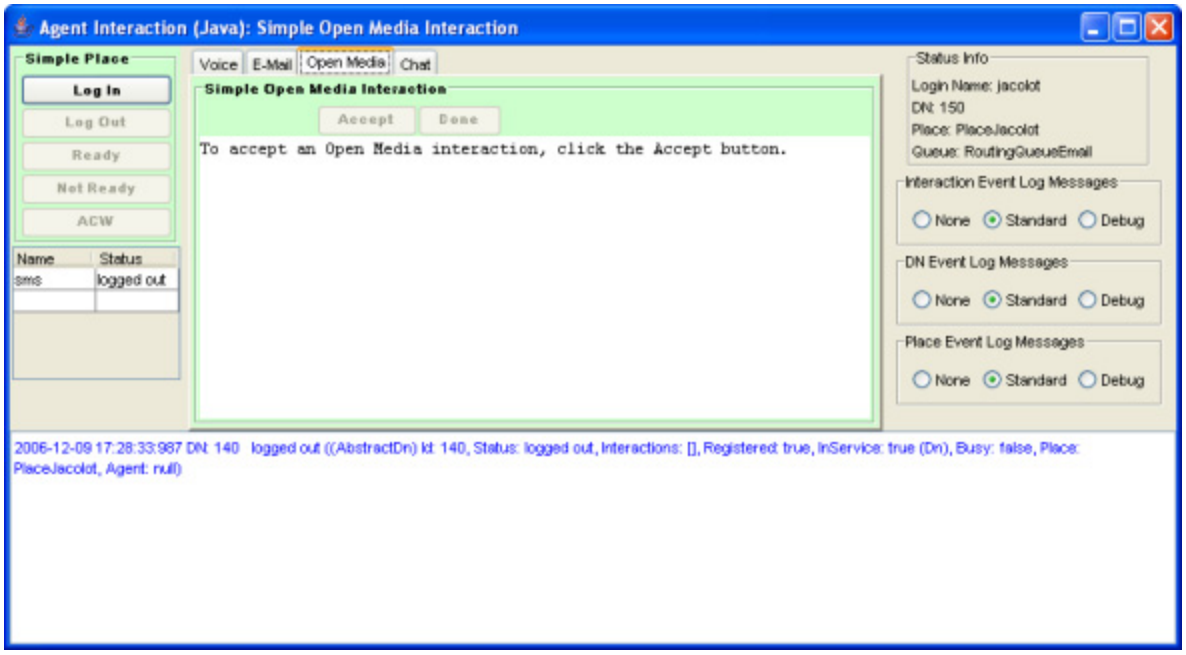
Handling a CRM Case Using Open Media

SimpleOpenMediaInteraction

This example extends `SimplePlace` and is very similar to `SimpleEmailInteraction`, which was explained earlier in this chapter.

> **Important**
>
> This example processes open media interactions. In order to run it, you will need to create your own open media interactions using the Media Interaction SDK. You must also set up a business process and routing strategy to get these interactions to your agent. The example is packaged with the business process and routing strategies that were used to develop it. If you use this business process and strategies, you will have to modify server names and Configuration Layer objects. For more information, refer to the Universal Routing documentation.

When you launch the application, there is a panel with two buttons: `Accept` and `Done`, as shown below. These buttons will let the agent accept an open media interaction and mark it done, respectively. When the agent accepts the interaction, information about it will be displayed in the GUI.



SimpleOpenMediaInteraction at Launch Time

Here are the steps you must take to create this application. As before, the steps that have already been done by `SimplePlace` will not be discussed here.

Set up Action Buttons

The `linkWidgetsToGui()` method is very similar to corresponding methods discussed earlier in this chapter. The two buttons are also very simple. Here is the action code for the `Accept` button, which is just like the `Accept` button for `SimpleEmailInteraction`:

```
sampleInteraction.answerCall(null);
```

Likewise, the `Done` button simply marks it done:

```
sampleInteraction.markDone();
```

Note that the `markDone()` method tells both Interaction Server and Universal Contact Server to stop processing the interaction.

These two sections of code are almost all that is new in this section of the example. Since the `setInteractionWidgetState()` code—used to synchronize the user interface—is very similar to what you have seen in previous examples, the only other thing to do is set up the event-handling code, which is also fairly simple.

Add Event-Handling Code

This example uses event-handling code that is largely similar to what you saw in `SimpleVoiceInteraction`.

> **Important**
>
> As explained in Threading section, you should write short and simple event handlers to avoid delaying the propagation of events.

As in `SimpleVoiceInteraction`, you check for a status of `TALKING` and for the correct media type. Then you read in the information and display it in the GUI:

```
//Checking that the event reports a change
//on an open media interaction
if(event.getSource() instanceof InteractionOpenMedia)
{
        //...
        // if the event involves the interaction to process,
        // it tests the interaction status
        if (sampleInteraction != null
                        && event.getInteraction().getId()==sampleInteraction.getId())
        {
                // When the open media interaction is in talking status,
                // it means that the agent or place is owner of the interaction
                // this is when you can process this interaction.
                if (event.getStatus() == Interaction.Status.TALKING) {

                        /// You can process the interaction.
                        /// In this example, processing the interaction corresponds
                        /// to displaying the text of the open media interaction
                        sampleInteraction=(InteractionOpenMedia) event.getSource();

                        // You can now access the Open Media interaction's content.
                        String someText = "Media type: "
                                + sampleInteraction.getOpenMediaType()
                                + "\nInteraction type: "
                                + sampleInteraction.getOpenInteractionType()
                                + "\nSubject: " + sampleInteraction.getSubject()
                                + "\nText: " + sampleInteraction.getText()
                                + "\nDate: "
+sampleInteraction.getDateCreated().toGMTString();

                        openMediaTextArea.setText(someText);
                }
        }
}
```