Platform SDK Developer's Guide

Interaction Server

6/9/2026

Interaction Server

You can use the Open Media Platform SDK to write Java or .NET applications that handle third-party work items in conjunction with the Genesys Interaction Server. You can also use it to work with servers that implement the Genesys External Service Protocol.

This document shows how to implement the basic functions you will need to write simple Interaction Server–based email applications. The first application is a simple media server that submits a new third-party work item. The second application enables an agent to receive a third-party work item, accept it for processing, and mark it done.

Java

Setting Up Interaction Server Protocol Objects

The first thing you need to do to use the Open Media Platform SDK is instantiate a Protocol object. To do that, you must supply information about the server you want to connect with. This example uses an `InteractionServerProtocol` object, supplying its URI, but you can also use name, host, and port information:

```
[Java]

InteractionServerProtocol interactionServerProtocol =
        new InteractionServerProtocol(
                new Endpoint(
                        InteractionServerUri));
```

After instantiating the `InteractionServerProtocol` object, you need to open a connection to Interaction Server:

```
[Java]

interactionServerProtocol.open();
```

Creating a Simple Media Server

The Open Media Platform SDK makes it easy to write a simple server that can submit third-party work items to Interaction Server. To write one, start by entering configuration information:

```
[Java]

// Enter configuration information here:
private String interactionServerName = "<server name>";
private String interactionServerHost = "<host>";
private int interactionServerport = <port>;
```

```
private int tenantId = 101;
private String inboundQueue = "<queue>";
private String mediaType = "<media type>";
// End of configuration information.
```

Now you will need to set up a protocol object:

```
[Java]

interactionServerUri = new Uri("tcp://"
        + interactionServerHost + ":"
        + interactionServerport);
InteractionServerProtocol interactionServerProtocol =
        new InteractionServerProtocol(
        new Endpoint(interactionServerName, interactionServerUri));
```

Once you have set up the protocol object, you can tell it the name of your application and let it know that it is a media server:

```
[Java]

interactionServerProtocol.setClientName("EntityListener");
interactionServerProtocol.setClientType(
        InteractionClient.MediaServer);
```

At this point, you can add user data associated with the new interaction:

```
[Java]

KeyValueCollection userData =
        new KeyValueCollection();

userData.add("Subject",
        "New Interaction Created by a Custom Media Server");
```

Now you can open the protocol object, and prepare the interaction to be submitted:

```
[Java]

try
{
        interactionServerProtocol.open();

        RequestSubmit requestSubmit = RequestSubmit.create(
                inboundQueue,
                mediaType,
                "Inbound");
        requestSubmit.setTenantId(tenantId);
        requestSubmit.setInteractionSubtype("InboundNew");
        requestSubmit.setUserData(userData);
```

If you use the `Request` method, you will receive a synchronous response containing a message from Interaction Server:

```
[Java]

Message response =
    interactionServerProtocol.request(requestSubmit);
System.out.println("Response: " + response.messageName() + ".\n\n");
```

Closing the Connection

Finally, when you are finished communicating with Interaction Server, you should close the connection to minimize resource utilization:

[Java]

```
interactionServerProtocol.close();
```

.NET

Setting Up Interaction Server Protocol Objects

The first thing you need to do to use the Open Media Platform SDK is instantiate a Protocol object. To do that, you must supply information about the server you want to connect with. This example uses an `InteractionServerProtocol` object, supplying its URI, but you can also use name, host, and port information:

[C#]

```
InteractionServerProtocol interactionServerProtocol =
        new InteractionServerProtocol(
                new Endpoint(
                        InteractionServerUri));
```

After instantiating the `InteractionServerProtocol` object, you need to open a connection to Interaction Server:

[C#]

```
interactionServerProtocol.Open();
```

Creating a Simple Media Server

The Open Media Platform SDK makes it easy to write a simple server that can submit third-party work items to Interaction Server. To write one, start by entering configuration information:

[C#]

```
// Enter configuration information here:
private string interactionServerName = "<server name>";
private string interactionServerHost = "<host>";
private int interactionServerport = <port>;
private int tenantId = 101;
private string inboundQueue = "<queue>";
private string mediaType = "<media type>";
// End of configuration information.
```

Now you will need to set up a protocol object:

```
[C#]

interactionServerUri = new Uri("tcp://"
        + interactionServerHost + ":"
        + interactionServerport);
InteractionServerProtocol interactionServerProtocol =
        new InteractionServerProtocol(
        new Endpoint(interactionServerName, interactionServerUri));
```

Once you have set up the protocol object, you can tell it the name of your application and let it know that it is a media server:

```
[C#]

interactionServerProtocol.ClientName = "EntityListener";
interactionServerProtocol.ClientType =
        InteractionClient.MediaServer;
```

At this point, you can add user data associated with the new interaction:

```
[C#]

KeyValueCollection userData =
        new KeyValueCollection();

userData.Add("Subject",
        "New Interaction Created by a Custom Media Server");
```

Now you can open the protocol object, and prepare the interaction to be submitted:

```
[C#]

try
{
        interactionServerProtocol.Open();

        RequestSubmit requestSubmit = RequestSubmit.Create(
                inboundQueue,
                mediaType,
                "Inbound");
        requestSubmit.TenantId = tenantId;
        requestSubmit.InteractionSubtype = "InboundNew";
        requestSubmit.UserData = userData;
```

If you use the `Request` method, you will receive a synchronous response containing a message from Interaction Server:

```
[C#]

IMessage response =
    interactionServerProtocol.Request(requestSubmit);
LogAreaRichTextBox.Text = LogAreaRichTextBox.Text
    + "Response: " + response.Name + ".\n\n";
```

Closing the Connection

Finally, when you are finished communicating with Interaction Server, you should close the connection to minimize resource utilization:

```
[C#]

interactionServerProtocol.Close();
```

Additional Topics

As support for the Platform SDKs continues to grow, new topics and examples that illustrate best-practice approaches to common tasks are being added to the documentation. For more information about using the Open Media Platform SDK, including functional code snippets, please read the following topics:

- Creating an Email - This article discusses how to use the Open Media and Contacts Platform SDKs in conjunction to create outgoing email messages. You can also apply the concepts illustrated here to other types of Interactions.