Agent Interaction SDK Java Developer Guide

Routing Points

12/13/2025

Contents

Routing Points

This chapter explains how to monitor routing points.

Routing Point Design

With the 7.*x* releases, the Agent Interaction (Java API) introduces visibility into route point and queue activity (based on the Genesys routing model).

The Universal Routing Server (URS) interprets strategies and routes interactions through routing points or routing queues before they are delivered to agents. Routing points can process interactions and provide results used by URS for further routing. For instance, a T-Server can host an Interactive Voice Response, which corresponds to a routing point and provides additional interaction data.

According to the type of routing point or queues through which the interaction goes, the treatment period varies. Agent Interaction (Java API) provides features to monitor the interaction activity on routing points:

- Monitoring a Routing Point/Queue that may be of the following types:
  - External Routing Point
  - Routing Point
  - Routing Queue
  - Virtual Routing Point
- Receiving events raised on these objects.
- Retrieving data of the `Interaction` that has been routed to the routing point.

Routing Point Information

To monitor routing points, Agent Interaction (Java API) provides access to DN routing point information and to the routing status of interactions processed on routing points.
The `DnRoutingPoint` interface registers to the T-Server and DN provides routing point information. With `DnRoutingPoint` methods, you can register `DnRoutingPointListener` listeners for getting `DnRoutingPointEvent` events, which describe status events like `in_service` or `out_of_service`.

The `RoutingInteraction` class describes an `Interaction` that arrived on a `DnRoutingPoint` instance. The`RoutingInteraction.Status` enumeration lists the possible interaction status on the routing point. For instance, if the routing interaction status is `IDLE`, it means the interaction is no longer on the Routing point, but it does not mean that the interaction is terminated (an agent may process it or another routing point may handle it.)

With `RoutingInteraction`methods, you can register `RoutingInteractionListener` listeners for getting `RoutingInteractionEvent` events, which notify changes on a routing interaction, that is, status or attached data changes.

Steps for Monitoring Routing Points

Now that you have been introduced to the routing point feature's design, it is time to outline the steps you will need to work with its events and objects.
In this section, monitoring routing points means monitoring status changes on the routing points and monitoring interactions on these routing points. the following code snippets shows how you can create
There are three basic things you will need to do:

- **Implement a DnRoutingPointListener listener** for getting status changes on routing points. This listener gets status changes for both DN routing points and routing point interactions. Here is how a `SimpleRoutingExample` class would do this:

```
public class SimpleRoutingExample implements DnRoutingPointListener {
        //...
        void handleDnRoutingPointEvent(DnRoutingPointEvent event)
        {
                DnRoutingPointEventThread p = new DnRoutingPointEventThread(event);
                p.start();
        }
        void handleInteractionEvent(RoutingInteractionEvent event)
        {
                RoutingInteractionEventThread p = new
RoutingInteractionEventThread(event);
                p.start();
        }
}
```

- **Implement the handleDnRoutingPointEvent() method** of your `DnRoutingPointListener` with a thread which updates your application's routing information. as shown here:

```
class DnRoutingPointEventThread extends Thread
{
        DnRoutingPointEvent event;

        public DnRoutingPointEventThread(DnRoutingPointEvent _event)
        {
                event=_event;
        }

        public void run()
        {
                DnRoutingPoint dnRoutingPoint = event.getDnRoutingPoint();
                System.out.println( “Event on DN RP: "+dnRoutingPoint.getId() + "
reason: "+ event.getReason().toString() + " new status: "+ event.getStatus().toString()
);
        }
}
```

- **Implement the `handleInteractionEvent()` method** (inherited from the `RoutingInteractionListener` interface) with a thread which updates your application with new interactions and interaction status changes for the monitored routing points.

```
class RoutingInteractionEventThread extends Thread
{
        RoutingInteractionEvent event;
```

```
        public RoutingInteractionEventThread( RoutingInteractionEvent _event)
        {
                event=_event;
        }

        public void run()
        {
                RoutingInteraction routingInteraction = event.getRoutingInteraction();
                //display new status is status change reported
                if(event.isStatusChanged())
                        System.out.println( "Ixn Event on ixn:
"+routingInteraction.getId() + " reason: "+ event.getReason().toString() + " new status:
"+ event.getStatus().toString() );
                //else the event reports extension changes
                else
                {
                        System.out.println( "Ixn Event on ixn:
"+routingInteraction.getId() + " reason: "+ event.getReason().toString() + " extension
changed: ");
                        // handle interaction extensions
                        //...
                }
        }
}
```

- **Register the listener for each Routing Point to be monitored** . For instance, the SimpleRoutingExample listener is added to a DN routing point in its constructor, as shown here:

```
public class SimpleRoutingExample implements DnRoutingPointListener
{
        //...
        public SimpleRoutingExample(
                        AilFactory ailFactory, String routingPointID)
        {
                try
                {
                        DnRoutingPoint dnroutingPoint = ailFactory.getRoutingPoint(
                                        routingPointID
                        );

                        dnRoutingPoint.addDnListener(this);

                }catch(RequestFailedException __e)
                {
                        System.out.println(__e.toString());
                }
        }
}
```

In your agent application, you can now create a SimpleRoutingExample instance for each routing point that you wish to monitor.