Service Management UI Help

Sample

12/14/2025

Sample

This sample is a Javascript Web interface, available through the Service Management UI.

- This sample illustrates how to implement a Desktop/Mobile browser web application that communicates with GMS and performs supported contact scenarios. It is primarily meant to be used by developers as a reference to build a Javascript-based web application with GMS.
- To enable the sample in the Service Management UI, configure enable-sample = `true` in your GMS configuration.

> **Important**
>
> You can also use this sample to test your GME deployment.

Access the Web demo of the Sample

Make sure that GMS is started. By default, the **samples** template is loaded and a **samples** service should be available in the list of **Services**.

To access the sample, start the Service Management UI and navigate through **Services and Tools > Tools > Sample**.

You can try a list of scenarios (1) by selecting a scenario, (2) click Connect to submit your query to GMS.

The following screens are available by clicking the corresponding tabs.

- `GMS` - The application home screen showing which GMS scenario can be executed.
- `Log` - Displays log messages related to client-server communication and application debug messages.
- `Queue` - For delay scenarios, checks the status of the interaction in the queue (when a request has been placed and is waiting for an agent).
- `Settings` - Application settings can be made on this screen.

Configure the Sample

[+] See the list of configuration steps

Resource Group—Add Access Number

Why:

GMS provides this access number to the user, and the user dials into this access number.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Tools > Resources.
2. Add the access number to the `DNIS` group.

GMS Service—Create Service request-interaction

Why:

This service is responsible for receiving the GMS request and providing an access number to the user.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Services > Configured Services.
2. Click `Add Service`.
3. Set `Configure Service = request-interaction`.
4. Set `Service Name = request-interaction`.
5. Click Save.

GMS Service—Create Service match-interaction

Why:

This service helps to match a voice call with an existing GMS service

responsible for providing the access number.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Services > Configured Services.
2. Click `Add Service`.
3. Set `Configure Service = match-interaction`.
4. Set `Service Name = match-interaction`.
5. Click Save.

GMS Service—Create Service request-access

Why:

This service lets you:

- Create a new access to a service
- Allocate a new DN in the resource group

How:

GMS Service Management UI

Procedure:

1. Go to the **GMS Service Management UI > Services > Configured Services**.
2. Click `Add Service`.
3. Set `Configure Service = request-access`.
4. Set `Service Name = request-access`.
5. Click Save.

Inbound SCXML Service—Voice

Why:

The inbound service matches the voice call with an existing GMS service. If a

matching service is found, the GMS user data is attached to the interaction, and the call is routed to the agent.

How:

- Configuration Manager > Switches > SIP_Switch
- Configuration Manager > Scripts

Procedure:

1. Create a route point associated with the access number configured in the procedure Resource Group Add Access Number.
2. Set Annex > Orchestration section > `application = script:GMSInbound.Voice.GMSMatchBuiltin`.
3. Create an enhanced routing script `GMSInbound.Voice.GMSMatchBuiltin`.
4. Set Annex > Application section > `url = http://<gmshost:gmsport>/genesys/1/document/service_template/callback/src-gen/IPD_Voice_GMSMatch.scxml`.
5. In Annex > `ApplicationParms`, set:
   `app_find_agent_timeout = 30`

   `app_match_gms_builtin = true`.

   `app_match_target = <target>` (Example: `Customer_Service@stat_server.GA`).

   `app_no_match_target = <target>` (Example: `All_Standard_Agents@stat_server.GA`).

   `app_require_access_code = false`.

   `app_require_ani = true`.

   `app_treatment_waiting_for_agent = <blank>` (A blank value will force the service to use a packaged music file.).
6. Make sure that MSML capabilities are configured and working to play treatments. This step is required because this service includes play treatments, and has a dependency on Media Server.

Interaction Workspace—Display GMS Attached Data

Why:

GMS attaches data to the call prior to routing it to the agent. This attached data is displayed to the agent when the call arrives at the agent desktop (Interaction Workspace), and helps the agent to understand the source of the call, as well as to understand the additional information sent from the customer's device when creating the Callback.

How:

Configuration Manager > Business Attributes

1. Create a new business `GMSCaseData` attribute of type `Interaction Operational Attribute`.
2. Create new attribute values:
   - `first_name`
   - `last_name`
   - `location_lat`
   - `location_long`
   - `GMS_Call_Direction`
   - `GMS_MatchMethod_AccessNumber`
   - `GMS_MatchMethod_ANI`
   - `GMS_MatchResult`
   - `GMS_MatchReason`
   - `GMS_ServiceName`
   - `GMS_UserData`
3. Set the following Application > InteractionWorkspace options:
   - `interaction-workspace > interaction.case-data.format-business-attribute = GMSCaseData`
   - `interaction-workspace > toast.case-data.format-business-attribute = GMSCaseData`

Implemented Scenarios

This sample supports the scenarios described in the Callback Scenarios. These scenarios are server-driven, which means that the server instructs the client with the actions needed to carry out the scenario. The client just needs to perform these actions and the follow-up dialog with the server. Therefore, the client is flexible enough to support any scenario that is built using the same kind of actions. The following actions are supported:

- `DialNumber` - The app makes a phone call when running on a mobile browser.
- `ConfirmationDialog` - The app displays a message requesting the user to confirm a follow-up action.
- `DisplayMenu` - The app displays a menu for the user to select an item that may affect how the scenario proceeds.
- `get-dialog*` - Retrieves the dialog details and displays the dialog to the user. Dialogs are limited to alerts.

This sample also supports the `request-interaction` scenario.

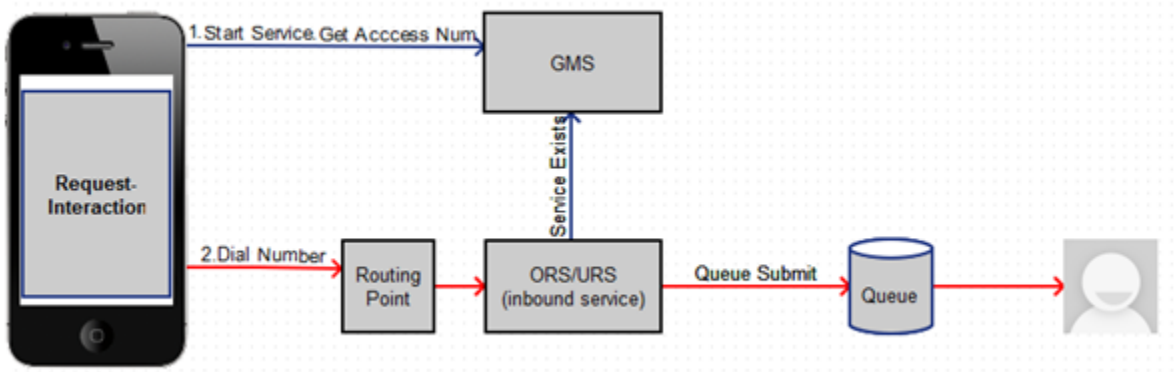
Push notifications through CometD are supported. Delayed scenarios are supported by using push notifications only; the app will not poll the server to be notified about agent availability.

Testing Built-in Scenarios

[+] See the instructions to test the built-in scenarios

> **Important**
> In the following scenarios, if the GMS Match fails, there will be no user data attached to the interaction.

Scenario request-interaction Test Procedure



1. On the Agent Desktop:
   - Log in agent.
   - Make voice ready.
2. Using the Javascript sample: Service Management UI > Lab > Sample:
   - Log in agent and make voice ready.
   - Set `Contact# = <customer phone from which call will be dialed>`
   - Set `Scenario = REQUEST-INTERACTION`
   - Click `Connect`.
   - Dial displayed Number to Call.
3. Expected result:

- Treatment is played.
- Call is routed to agent.
- Toast is displayed with attached data.
- Call is connected to agent.
- For a successful GMS call, `GMS_MatchResult = SUCCESS` is displayed in the agent desktop as attached data.

Compiling and Running the Sample

> ### Important
> This step is required only if you download the code sample in order to modify the source code.

Prerequisites

In order to use this sample app, you need to have GMS installed and running, and the services that you want to make use of must be deployed. The source code of this sample is available via a downloadable zip file: Genesys Mobile Services JavaScript Sample Zip File

Access the Sample

1. Download and unzip the .zip file from the above link.
2. Check the **readme.md** file in the unzipped files for the details on starting the sample and configuring it.

> ### Important
> - The CometD client is automatically started when the application loads in the browser.
> - Make sure that your URL starts with the value specified in GMS > Server > `external_url_base` when you access the Service Management UI.

About the Code

The majority of the code is in two files:

- `index.html` - Controls the presentation aspects of the application, which includes the GMS response

handler.

- `gms.js` - Responsible for interfacing with GMS and as well as managing the CometD connection.

index.html

The following screens are presented to the user and can be displayed by clicking the corresponding tabs.

- `GMS` - The application home screen showing which GMS scenario can be executed.
- `Log` - Displays log messages related to client-server communication and application debug messages.
- `Queue` - For delay scenarios, checks the status of the interaction in the queue (when a request has been placed and is waiting for an agent).
- `Settings` - Application settings can be made on this screen.

gms.js

Two objects are implemented in this file:

- `gmsInterface` - Allows the creation of GMS callback services and delegates responses to `index.html::onResponseRecieved`.
- `gmsNotificationClient` - Responsible for starting the CometD client and connecting to the GMS CometD channel. When the message is received, the callback function `index.html::onCometNotification` is invoked.

Disclaimer