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.

GVP Deployment Guide

How the Supplementary Services Gateway Works

5/1/2025

How the Supplementary Services Gateway Works

Read here about how the Supplementary Services Gateway (SSG) performs its role in a GVP deployment:

Operational Overview

The Supplementary Services Gateway receives requests for outbound-call initiation from third-party Trigger Applications (TA). The requests are validated and placed in persistent storage in the Supplementary Services Gateways external database. If the outbound call succeeds (or fails after specified number of retries), the Supplementary Services Gateway notifies the trigger application with a result notification URL in the form of an HTTP request (which the trigger application includes in the call initiation request).

Embedded HTTP Server

The Supplementary Services Gateway has an embedded HTTP server which communicates with the trigger application to service HTTP GET, POST, and DELETE requests.

The embedded HTTP server supports both HTTP and HTTPS over Secure Socket Layer version 1 (SSLv1), SSL version 2 (SSLv2), and Transport Layer Security version 1 (TLSv1).

The default page or identifier for the embedded HTTP server is SSG, and is configured in Genesys Administrator with the HTTPDefaultPage parameter in the http section of the Supplementary Services Gateway Application object. HTTP or HTTPS URIs targeted for the Supplementary Services Gateway use the following format:

```
http(s)://<ssg host>:<http/https port>/SSG?...
```

Secure Communications

The Supplementary Services Gateway also supports HTTPS for secure communication.

Outbound-Call Establishment

When the Supplementary Services Gateway receives an HTTPS POST trigger from the trigger application with CreateRequest in the body, it initiates an outbound call by sending a TMakePredictiveCall request to SIP Server. SIP Server establishes two call legs; one to Media Control Platform through the Resource Manager and one to the external party. The call legs are then bridged, which invokes a third-party call-control scenario.

Requests for Service

When the request reaches the Resource Manager it is treated like any other request for service within GVP. The Resource Manager determines the type of service and application profile that will be used to fulfill the request and sends it to the Media Control Platform, which then provides the media-centric services for the Supplementary Services Gateway specifically VoiceXML applications. See also, Call Initiation Through SIP Server.

Component Management

Like all other GVP components the Supplementary Services Gateway is managed (stopped, started, or restarted) and monitored by the Genesys Administrator web interface. It also receives configuration and provisioning information from the Configuration Server.

Requests and Responses

The Supplementary Services Gateway and the trigger application use the HTTP request/response standard used in server/client environments. The Supplementary Services Gateway sometimes acts as the server and sometimes as the client, depending on whether it is requesting or providing information. (See Supplementary Services Gateway Interfaces).

This section describes the following types of requests and the corresponding responses:

- Create Requests
- Query Status Requests
- Cancel Requests

Create Requests

The Supplementary Services Gateway receives HTTP triggers from the trigger application for single and bulk outbound-call requests. The call triggers include a Token which uniquely identifies the trigger application submitting the request. The Supplementary Services Gateway responds to these call triggers by generating a RequestID and managing the status of outbound-call requests.

Outbound Requests HTTP POST

TAs use the HTTP POST method to submit outbound-call requests to the Supplementary Services Gateway. HTTP POST is used to create, query, and cancel requests. The body of a single POST can contain a single CREATE, QUERY, or CANCEL request, multiple (bulk) CREATE, QUERY, or CANCEL requests, or any combination of all three requests. The Supplementary Services Gateway does not impose a limit on the size of an HTTP POST request.

Request Validation

The Supplementary Services Gateway validates each HTTP POST request sent from the TAs based on the following criteria:

- The body of each HTTP POST request must conform to the schema that is defined for the HTTP POST method.
- Each POST request must include the TenantName parameter in the query string of the POST Request URI for example,
  `<host>:9800/SSG?TenantName=<Tenant_Name>`
- The Content-Type for each request must be text/XML or application/XML.

If the POST request meets this criteria, it is validated and added to the Supplementary Services Gateways persistent storage (an external database).

Request Acceptance

After validation, the Supplementary Services Gateway parses the XML data in the POST request body and generates a unique identifier (RequestID) for each request. The RequestID is stored in the Supplementary Services Gateways database along with the request. The Supplementary Services Gateway sends the RequestID and Token back to the trigger application to indicate that the request has been accepted and simultaneously processes each request in storage. A detailed description of this process is described in Basic Outbound-Call Flow on page 464.

Each HTTP POST request must adhere to the XML schema. The CreateRequest part of the POST body must include certain mandatory attributes, such as: Token, IVRProfileName, Telnum, NotificationURL, MaxAttempts, and TimeToLive. For example:

```
<CreateRequest Token="Token" MaxAttempts="2" TimeToLive="123s"
IVRProfileName="Application" Telnum="9884719189"
NotificationURL="http://182.123.12.12/DIR/OutURL.xml" Ani="12345"></CreateRequest>
```

For a detailed description of the attributes of the CreateRequest, see the Genesys Voice Platform 8.5 User's Guide.

Responses to Outbound Request HTTP 200 OK

In most cases, the Supplementary Services Gateway responds to requests from the trigger application with HTTP 200 OK responses. The response section of the 200 OK message can contain single or bulk requests, depending on whether the POST (to which it is responding) is a single or bulk request. The response contains the RequestID, the Token, and the request result (success or failure)

which is in a format defined by the XML response schema. For more information about the XML response schema, see the Genesys Voice Platform 8.5 User's Guide.

Success and Failure Response Types

The Supplementary Services Gateway generates responses to indicate success or failure, based on the following methods of validation:

- If a request is successful, the response sent to the trigger application includes the RequestID and Token for the request, or for each request within the bulk request with a SUCCESS response type. The trigger application must include the RequestID in any further requests (such as, status querying or request cancellation).

- If a request fails during request validation or when it is being stored in the external database, a FAILURE response type is sent within the 200 OK response, along with the Token, a Reason Code, and Reason. See the following SSGResponse part of a 200 OK bulk response:

```
<SSGResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<ResponseElement ResponseType="SUCCESS" Token="T1001" RequestID="123435"/>
<ResponseElement ResponseType="SUCCESS" Token="T1002" RequestID="123436"/>
<ResponseElement ResponseType="FAILURE" Token="T1003" ReasonCode="120" Reason="DB
Insertion failed"/>
</SSGResponse>
```

- If parsing or validation fails for a bulk POST request, the entire request fails and the Token is not passed back to the trigger application. However, if it is parsed and validated successfully, but later encounters an operational error, (for example, a request fails when it is added to the database) the bulk response contains a mix of SUCCESS and FAILURE response types and the Tokens for each request are passed back to the TA.

- If a POST request contains a mixture of CREATE, QUERY, and CANCEL requests, and the number of database records is approaching the maximum during processing, the Supplementary Services Gateway inserts them into the database until the maximum threshold is reached and then, executes the QUERY and CANCEL requests. The Supplementary Services Gateway sends a FAILURE response for the remaining requests with a Reason Code, and Reason.

The Supplementary Services Gateway also generates HTTP 500 responses to indicate internal errors. For a detailed description of the attributes of the SSGResponse part of the response and a complete list of status codes in the response, see the Genesys Voice Platform 8.5 User's Guide.

Query Status Requests

The trigger application uses two methods to query the status of previously submitted requests that are stored in the Supplementary Services Gateway database: HTTP GET for a single query and HTTP POST for single or bulk queries. When the trigger application sends GET or POST query requests, they must contain the RequestID and TenantName parameters. The request is passed on in the HTTP GET query string, or in the body of the HTTP POST request. In the HTTP POST query request, the Content-Type must be text/xml or application/xml and must conform to the Supplementary Services Gateway XML schema.

Responses to Status Queries

The following responses are the same for both the GET and POST methods of querying requests:

- If the Supplementary Services Gateway finds the request (or each request if POST is used) in its database and obtains the status of the request, a 200 OK response, with a SUCCESS ResponseType is sent to the trigger application. The response also contains the RequestID, Token, and other attributes (see the example in this section).

The Content-Type in the 200 OK response is text/xml. If the Content-Type of the POST request is neither text/xml or application/xml, the Supplementary Services Gateway returns a 415 status code in the body of the response, indicating an invalid content type was used.

- If query request parsing fails, mandatory attributes are missing, or database operation fails, only the RequestID, ReasonCode, and Reason parameters are passed back to the TA.
  - If validation or parsing fails, the entire POST or GET request fails and the Supplementary Services Gateway generates a ReasonCode and Reason (or failure description).
  - If validation and parsing succeeds, but there are other failures for example, a specific RequestID is not found in the database the RequestIDs are passed back in the 200 OK response.

The following is an example of the SSGResponse part of a 200 OK bulk query response:

```
<SSGResponse>

<ResponseElement ResponseType="SUCCESS" Token="T1001"RequestID="123435"
TenantName="Environment" IVRProfileName="Application" Telnum="11011"
NotificationURL="http://182.24.129.82/Dir/Response.asp" AttemptsMade=4
MaxAttempts=7 TimeToLive="12000s" TTLRemaining="3477s"Status="Waiting to be processed"/>

<ResponseElement ResponseType="FAILURE" RequestID="1234" ReasonCode="404"
Reason="RequestID not found in the Database"/>

</SSGResponse>
```

For a complete list and description of the HTTP request and response attributes, see the Genesys Voice Platform 8.5 User's Guide.

Cancel Requests

Trigger applications use two methods to cancel pending outbound requests: HTTP DELETE to cancel a single request and HTTP POST to cancel a single or bulk request. When the trigger application sends DELETE or POST requests, they must contain the RequestID and TenantName parameters. The request is passed on in the HTTP DELETE query string, or in the body of the HTTP POST request. The Content-Type of the HTTP POST cancel request must be XML text or an XML application that conforms to the XML schema.

Responses to Cancel Requests

A request that is not in progress (the TMakePredictiveCall request has not been sent to SIP Server), can be cancelled and the record deleted from the Supplementary Services Gateway database. However, if the request is already in progress, the Supplementary Services Gateway does not

attempt to delete the request from its database, but sends a FAILURE response type in the 200 OK response.

The responses to requests for the DELETE and POST methods of cancellation requests are the same as for query requests. See Responses to Status Queries.

The following is an example of the SSGResponse part of a 200 OK bulk cancellation response:

```
<SSGResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ResponseElement ResponseType="SUCCESS" RequestID="1121245"/>
<ResponseElement ResponseType="FAILURE" RequestID="1000000"
ReasonCode="106"Reason="Invalid RequestID"/>
</SSGResponse>
```

Asynchronous Result Notifications

The Supplementary Services Gateway notifies the trigger application about the final outcome of the outbound request in the form of a NotificationURL, which is a mandatory parameter in the CreateRequest. The Supplementary Services Gateway performs an HTTP GET on the Notification URL and appends certain query-string parameters that indicate the status of the outbound request

Asynchronous Result Notification Success

When a call is made through a trunk group DN and a TreatmentApplied event is received from the SIP Server, the Supplementary Services Gateway deems the call successful, appends the query string, and sends the Notification URL to the trigger application.

Notification URL Query-String Parameters

Among the query-string parameters that are passed back to the trigger application in the Notification URL is the CallUUID, which is a unique ID that is generated by the SIP Server for the call. When the outbound call has been attempted multiple times, only the last calls UUID is sent in the Notification URL. In addition, the Status parameter contains the reasons (separated by a colon [:]) that each prior attempt failed for example,

```
AnsweringMachineDetected:RingNoAnswer:DestinationBusy
```

The following is an example of a Notification URL that is sent when the outbound call is successful:

```
//test.genesyslab.com/trigger/
result.asp?Token=T3001%RequestID=123345&TenantName=Environment&IVRProfileName=Application&Telnum=
8GQE8C05B56SV2HRTTJ8M9RFR8000001%Result=SUCCESS&Status=DestinationBusy:RingNoAnswer
```

Asynchronous Result Notification Failed

Requests can fail at two stages of the process: before they are stored in persistent storage and after they have been processed. An outbound request fails after processing for various reasons: It

exceeded the maximum attempts (MaxAttempts in the request), the time-to-live (TTL) expired, or a permanent error caused it to fail.

The query-string parameters that are passed back to the trigger application in the Notification URL are the same as for a success notification, and the Status parameter contains the reasons (separated by a colon [:]) that each prior attempt failed. For example,

```
ExternalError:RingNoAnswer:DestinationBusy:MaxAttempts Exceeded
```

Call Initiation Through SIP Server

The Supplementary Services Gateway receives HTTP requests from the trigger application along with parameters in the body of the POST requests. Acting as a T-Lib client, the Supplementary Services Gateway uses SIP Servers T-Lib interface to send a TMakePredictiveCall, with the parameters sent as extensions. SIP Server then establishes two call legs by sending one INVITE request to the Media Control Platform (through Resource Manager) and another to the external party.

SIP Server, acting as a T-Server, establishes two call-legs one to the Media Control Platform through the Resource Manager and one to the external party. The call legs are then bridged, invoking a third-party call. The call leg that is established to the Media Control Platform, contains parameters that are passed on in the INVITE messages to Resource Manager as Request URI parameters or special headers. The Resource Manager uses these parameters to determine if a specific IVR Profile is required. After the call legs are established, the VoiceXML application associated with the IVR profile is played for the external party.

Connection to SIP Server in HA Mode

To Supplementary Services Gateway establishes a connection with SIP Server in HA mode by obtaining the configuration details for both the primary and secondary SIP Server through CCILib.

After the connection to the primary SIP Server is configured in the Supplementary Services Gateway Application (in the Connections tab), it uses the port ID in the primary SIP Server to establish a connection with the secondary SIP Server. It attempts to establish a connection with the primary SIP Server first. If there is no response, it tries with secondary SIP Server by using a simple round robin mechanism.

Secure Connections Through TLS

SIP Server can connect to the Supplementary Services Gateway on any of its secure or unsecured ports. Secure ports are configured by creating security certificates, which enable the Supplementary Services Gateway to interact with SIP Server by using TLS.

For a procedure that describes how to create security certificates, see "Chapter 3: Configuring Common Features" in the Genesys Voice Platform 8.5 User's Guide.

DNs for Resource Manager and External Party

The Resource Manager trunk-group DNs are registered on the Supplementary Services Gateway in the Tenant1 section and contains the following parameters, including the mandatory TGDN

parameter:

```
[Tenant1]
TGDN=<tg-dn>
RPDN=<rpdn>
AccessGroup=<grp-name>
DialPrefix=<DialPrefix-name>
```

(The value that is configured in the TGDN parameter is used as the tenant name.)

The external-party Trunk DNs are configured on the SIP Server. SIP Server selects the external-party DN, based on the mandatory telnum parameter, which is one of the parameters in the body of the HTTP request.

Call-Progress Detection

The Supplementary Services Gateway and SIP Server support Call-Progress Detection (CPD) on the Media Gateway or on the Media Server module of the Media Control Platform. CPD is optional, however if it is configured on both the Media Gateway and Media Server, the Media Gateway takes precedence as the CPD provider. SIP Server receives the CPD result from the CPD provider and passes it back to the Supplementary Services Gateway through a T-Event.

CPD Control Parameters

Trigger applications send CPD control parameters to the Supplementary Services Gateway in the CreateRequest part of the HTTP POST requests. All of the CPD parameters are optional. The value of the record parameter specifies if the CPD part of the call is recorded `true` (or 1) if the CPD is to be recorded and `false` (or `0`) if it is not. The value of the preconnect parameter is mapped to the cpd-on-connect extension in the TMakePredictiveCall request and specifies when to start CPD `true` if CPD is to be started when the first packet is received, and `false` if CPD is started when the call is connected. If there are no CPD parameters in the CreateRequest part, SIP Server relies on its own CPD configuration.

For a complete list and description of CPD parameters, see the Genesys Voice Platform 8.5 User's Guide.

The following is an example of the CreateRequest part of a POST request with CPD parameters:

```
<CreateRequest Token="Token" MaxAttempts="2" TimeToLive="123s"
IVRProfileName="Application" Telnum="9884719189"
NotificationURL="http://182.123.12.12/DIR/OutURL.xml" Ani="12345">

<cpd record="false"
postconnecttimeout="6000ms"
rnatimeout="60s"
preconnect="true"
detect="all"/>

</CreateRequest>
```

For information about how the Media Control Platform (Media Server) supports CPD, see How the Media Control Platform Works.

Port-Availability Notifications

The Supplementary Services Gateway relies on T-lib event notifications from SIP Server to receive notification of available ports. The Resource Manager and Media Control Platform have a finite number of ports available for outbound calls; therefore, the Supplementary Services Gateway must be aware of the maximum number of ports that are available for a tenant and the number of ports that are available at any given time.

Subscription Request

The Supplementary Services Gateway sends the subscription request to SIP Server as a RequestRegisterAddress extension in the resource DN (configured in the Supplementary Services Gateway). SIP Server sends notifications on behalf of the DN using the EventResourceInfo event with extensions that include the total number of ports that are allocated and the total number of ports that are available for outbound calls. If the Resource Manager is unavailable, SIP Server returns zeros (0) for total-ports and available-ports parameters.

The Resource Manager provides port-availability notifications that include data that is specified for each tenant. When the Supplementary Services Gateway receives these notifications for a tenant trunk group, it uses these new values to manage the tenant campaigns.

Persistent Storage

The Supplementary Services Gateway uses an SQLite external database for its persistent storage, supported on Windows and Linux. It stores requests for outbound-call initiation persistently for the following reasons:

- Although a limited number of Resource Manager and Media Control Platform ports are available for outbound calls, outbound requests are not rejected, because they can be stored and executed as ports become available.
- Previous requests can be retrieved from the database across Supplementary Services Gateway restarts. Incomplete calls are reinstated as new calls and re-attempted.
- The progress and status of each request can be maintained in the database.

There is a threshold for the maximum number of records allowed in the database. When the maximum number of records is reached, further requests from trigger applications are rejected with the appropriate error code in the response. See also, Responses to Outbound Request—HTTP 200 OK.

Processing Requests

The Supplementary Services Gateway processes HTTP requests in batches based on information received in EventResourceInfo messages sent every five (5) seconds from SIP Server. The EventResourceInfo messages contain the total number of GVP ports, the number of available GVP ports, and the TenantName.

Fetching from Database to InMemQ

Requests are fetched from the database and placed in the Supplementary Services Gateway in-memory queue (InMemQ). The number of fetched requests can be configured with the Request Batch Size parameter using one the following values:

- `TotalPorts` Use the total ports returned in the EventResourceInfo message from SIP Server. This is the default value.
- `AvailablePorts` Use the available ports returned in the EventResourceInfo message from SIP Server.
- `NNN` A specified number, for example, 610.

When the InMemQ falls below a certain percentage of the Request Batch Size parameter the next database fetch cycle is triggered and the state of each fetched record is marked INITIATED in the database. The default percentage is 25% and can be configured using the Queue Low Watermark parameter.

Allocation of InMemQ for IVR Profiles

A separate InMemQ exists for each IVR Profiles, the contents of which is processed using a round-robin algorithm. If there are requests in the database for more than one IVR Profile, a portion of the batch size is allocated to each application. The Application Slot Calculation parameter controls how the batch size is divided using one of two values:

- **PROPORTIONATE** The batch size is divided in the same proportion as the number of requests pending for the IVR Profiles. This is the default value.
- **EQUAL** The batch size is divided equally among the IVR Profiles.

Prioritizing Requests

When requests are fetched from the database for a specific IVR Profiles, they are prioritized in the following ways:

- Requests with earlier `NextRetryTime` values are picked up ahead of those with later NextRetryTime values.
- If requests have the same `NextRetryTime` values, the requests with least remaining time-to-live (TTL) have priority over those with higher remaining TTL.
- If requests have the same `NextRetryTime` values and same remaining TTL, the requests with the higher number of attempts have priority over those with fewer number of attempts.

Each IVR Profile can have a combination of new requests in the database and requests that were already processed (the call failed and they will be re-attempted). The value of `EqualPriorityToNewAndOld` determines how the Supplementary Services Gateway picks up

requests, for example, if the value is:

- **False** The requests for each IVR Profile are picked up based on the NextRetryTime, remaining TTL, and number of attempts. This is the default value.
- **True** For each IVR Profile in a fetch cycle an equal amount of new and previously-attempted requests are picked up.

Processing Requests from InMemQ

The requests in InMemQ are passed to the SIP Server to make outbound calls. The Supplementary Services Gateway checks the AvailablePorts parameter in the EventResourceInfo message to determine the number of GVP ports available for outbound calls. The number of calls the Supplementary Services Gateway is able to pass to SIP Server is determined by the value of the PortLoadFactor parameter, which has an nnn value (expressed as a percentage of the available ports). The default value is 100.

After the number of outbound calls is determined, the Supplementary Services Gateway throttles them rather than send them to SIP Server all at once (which could result in high Calls Available Per Second [CAPS] and overload the components). The throttling functionality is controlled by the value X of the pacing.caps.CallRequestsPerSecond parameter that the Supplementary Services Gateway sends to SIP Server. The Supplementary Services Gateway only attempts X calls per second to SIP Server. It continues to make X calls every second until the number of requests that are configured in the PortLoadFactor parameter is exhausted.

Error Handling

Failed requests are categorized as permanent or temporary failures and handled in the following ways:

- Permanent failures The record is marked as PROCESSED in the database, the result is marked as FAILURE, and the deleteflag is set.
- Temporary failure, where either the number of attempts exceeds the maximum number of attempts (#Attempts>MaxAttempts) or the TTL has expired The record is processed the same as for a permanent failure.
- Temporary failure that have attempts left and TTL remaining The NextRetryTime is calculated:
  - For internal failures (within GVP), the number of attempts is not updated in the database. Requests are pushed either to the back or in front of the InMemory Queue. The NextRetryTime value is calculated based on the NextRetryInterval parameter value. The default value is 10 seconds.
    - For external failures, the number of attempts is increased in the following ways:
    - For Busy or SIT Tones (other than those that are treated as permanent failures) The NextRetryTime parameter value is paced closer together and spread out later.
    - For all other external failures (such as Answering Machine and Ring No Answer) The NextRetryTime parameter value is paced equally. (The remaining TTL is divided equally among the remaining attempts.)

After the NextRetryTime parameter value is calculated, the request is either left in the InMemQ (if the current time >= NextRetryTime) or put back into the database and processed later.

Calls that succeed are marked as PROCESSED in database, the result is marked as SUCCESS, and the deleteflag is set.

Database Cleanup

The persistent storage database is monitored and cleaned up at regular intervals (configured with the `Clean Interval` parameter; default value = 180 [seconds]). Cleanup occurs in three stages:

- The database is trawled, and all records that have `deleteflag` set are collected.
- A SUCCESS or FAILURE Notification URL is sent to the trigger application for each record. If an error occurs during invocation of the Notification URL, an SNMP trap is generated.
- The selected records are deleted from the database.