# Bot Gateway Server Quick Start Guide

Bot Gateway Server Current

5/22/2025

# Table of Contents

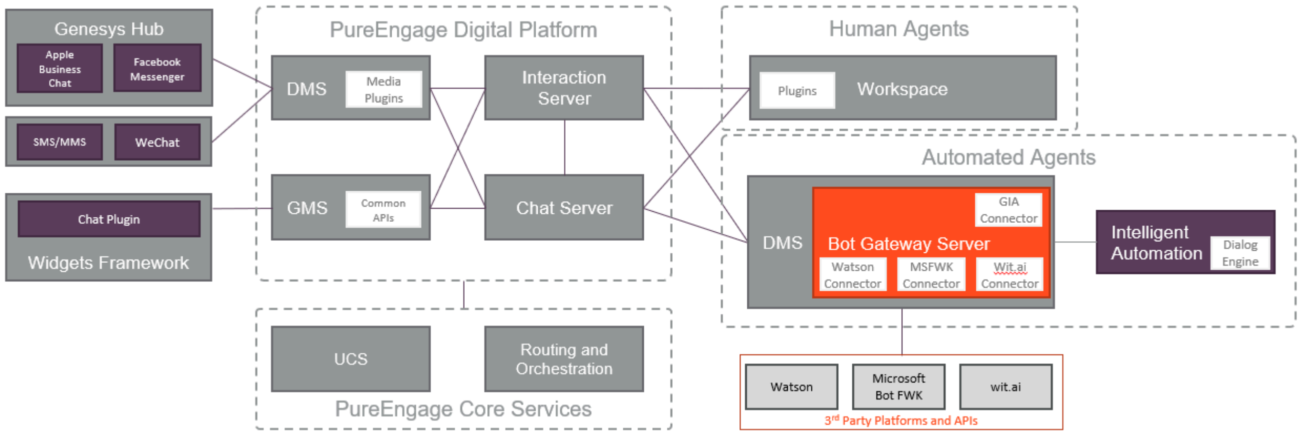
**Bot Gateway Server Administration** **3**
- How Bot Gateway Server works 5
  - ESP methods 9
- How to deploy Bot Gateway Server 13
  - GAX deployment 20
  - Integrating BGS with Genesys Historical Reporting 22
- How to develop a new bot for Bot Gateway Server 31

# Bot Gateway Server Administration

Bot Gateway Server (BGS) is a Genesys component which is implemented as an eServices Digital Messaging Server (DMS) driver. BGS provides an integration platform for deployment of various chat bots that participate in chat sessions conducted by Chat Server.

For more context, watch our quick video with some information you might find helpful:

## How Bot Gateway fits into the Genesys Engage architecture



## Why we need bots

Whenever a user connects via chat, SMS, a messaging app, or Facebook Messenger, a chat session is established and routed to a human agent. During the session, which is handled by Chat Server, the human agent uses Workspace as the interface to receive and respond to messages from the user.

Think of BGS as the equivalent of Workspace for automated agents (bots). BGS maintains connectivity during a chat session between Chat Server and the bot. BGS sends user input to the bot and receives and forwards the bot response back to the user. BGS works with and without human agents being involved in the conversation.

We expect customers to use bots, and thus BGS, to resolve common customer experience use cases in the following ways:

- Self-Service - Resolving customer issues without the need for a human agent to get involved.
- Natural Language Detecting - Rejecting comments that are in violation of profanity rules; determining sentiment or analyzing the intent of a conversation to route a user to the right bot or human agent.
- Agent Assist - Helping to guide a human agent through a conversation by delivering relevant

information in the context of the interaction.

- User Assist - Helping users with translations or PCI-compliant credit card transactions.

See the following topics for more information:

- How Bot Gateway Server works
- How to deploy Bot Gateway Server
- How to develop a new bot for Bot Gateway Server

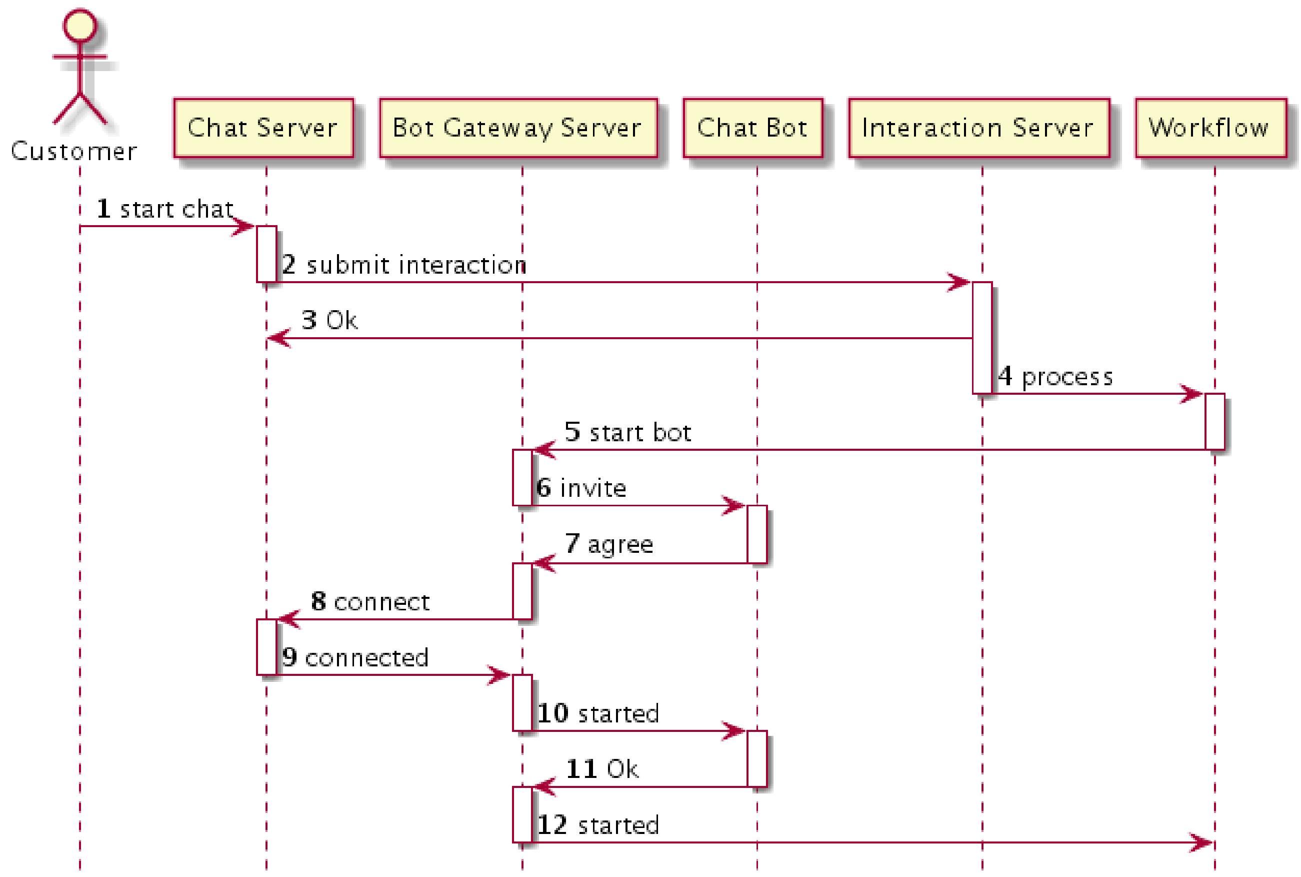
# How Bot Gateway Server works

Bot Gateway Server (BGS) is a Genesys component that is implemented as an eServices Digital Messaging Server (DMS) driver. BGS provides an integration platform for the deployment of chat bots that participate in chat sessions conducted by Chat Server. At a high level, the functional behavior of BGS is as follows:

- BGS is deployed as dedicated instances of DMS. Chat bot plugins (packaged as JAR files) are placed into a special subfolder of BGS and must be configured within the options of the DMS application.
- A workflow controls chat bots by sending an ESP request through Interaction Server. Workflows can start and stop the bot, as well as send data to the bot. For example, to connect a bot to chat session:

1. The Workflow sends ESP request **StartBot** to the BGS application.
2. BGS searches for the bot ID and, if found, requires the bot plugin to create a bot instance that serves this particular chat session.
3. After obtaining the bot instance, BGS connects the chat bot instance to the chat session as a user of type agent with a special attribute signifying "bot" style, and relays the notifications about chat session activity to the bot instance. The chat bot instance can use the BGS API to submit messages into the chat session, to exit chat session, and to update the userdata of the interaction.
4. At the end (either defined by a condition or caused by a **StopBot** ESP command) BGS removes the chat bot instance from the chat session and disposes the chat bot instance.

The following diagram explains how the chat bot is connected to the chat session:



## High Availability support

Beginning in version 9.0.006.04, Bot Gateway Server supports high availability mode which provides the following functionality:

- Automatic restart of the bot in case of a BGS switchover (either manual/planned or automatic due to failover). A special API method allows the bot to save and restore the bot context.
- Automatic reconnection to a chat session after losing the connection to Chat Server.

## Workflow ESP API description

BGS ESP API (the control API) is intended to be used from a workflow (strategies of URS/ORS) to start/stop a bot and to send data to a bot. Besides using the API in a workflow, you can use the API from other applications like Custom Agent Desktop and other bots. The BGS installation package includes a sample URS and ORS workflow (see the description on how to test the bot on the Deployment page).

In order to send the command or request to BGS, the strategy must use the **External Service** object and provide the following:

- **Application Type:** `SocialMS`
- **Application Name:** *The name of your DMS application where BGS is deployed*
- **Service name:** `ChatBotPlatform`
- **Method:** Choose one of the 4 ESP methods

## How chat bots integrate with the workflow

Chat bots integrate with the workflow using the following two modes:

- Waiting mode: To use this mode, the `ChatBotHoldup` startup parameter must be set to `true`. In order to facilitate the workflow to hold the interaction until the bot finishes, a special workflow queue view with a scheduling condition must be used which is demonstrated in the workflow sample.
- Parallel mode: The workflow immediately routes the interaction to an agent after the bot starts. The bot works in parallel with the agent.

Tip

The Bot Gateway Server installation package (IP) provides a sample workflow that demonstrates both modes.

## How chat bots finish execution

BGS stops the execution of a given chat bot instance and removes it under the following conditions:

| Condition | BGS removes bot from chat session |
|---|---|
| When **StopBot** ESP request is received. | with after action **keep-alive** |
| When all participants (except bots, system, and external users) have left the chat session. | with after action **keep-alive** |
| When a bot instance explicitly requests (in Java API) to leave a chat session. | with after action **keep-alive** |
| When an agent joins a chat session (after the chat bot was already connected to the chat session). This condition applies only if the parameter **StopBotOnAgentArrival** has a value of `true`. | with after action **keep-alive** |
| When a customer leaves a chat session (after the chat bot was already connected to the chat session). This condition applies only if the parameter **StopBotOnCustomerLeft** has a value of `true`. | with after action **keep-alive** |

| Condition | BGS removes bot from chat session |
|---|---|
| When Chat Server removes the bot participant because:<br><br>• Another participant explicitly requested to remove the bot<br>• The chat session is ending due to the following conditions:<br>  • Agent left the chat session with after-action **close** if no agents or **force close**<br>  • Idle control monitoring.<br>  • Interaction stopped when only customer was present in the chat session.<br>  • Unrecoverable issue with UCS in HA (high-availability) mode. | n/a |
| When BGS detects a disconnection from Chat Server, or a shut down of BGS has started. | n/a |

# ESP methods

**Warning**

It is vitally important to send userdata in every External Services Protocol (ESP) request as it contains several important parameters (such as the interaction ID) required by BGS to identify and start a bot. Both Interaction Routing Designer (IRD) and Composer contain an attribute of ESP block where sending userdata can be enabled.

## StartBot

| Parameter | Default value | Description |
|---|---|---|
| ChatBotID | (hardcoded) | Required.<br><br>The ID of the BGS bot plugin (this ID is hardcoded inside the bot and returned by `getBotId()`). The combined length of ChatBotID and ChatBotName must be less than 49 characters. |
| _umsChannel | (empty string) | Required (see General notes).<br><br>Value `channel-chatbot` must be provided. |
| ChatBotName | (empty string) | Optional.<br><br>The name of the "external" bot (if the bot plugin implements a connector to other bot frameworks). The combined length of ChatBotID and ChatBotName must be less than 49 characters. |
| Nickname | (empty string) | Optional.<br><br>Specifies how the bot will be presented in a chat session. |
| Visibility | ALL | Optional.<br><br>Possible values are ALL, INT, VIP (see General notes). |
| StopBotOnAgentArrival | false | Optional.<br><br>If `true`, BGS stops the bot when the first agent arrives in the chat session. |
| StopBotOnCustomerLeft | false | Optional. |

| Parameter | Default value | Description |
|---|---|---|
| | | If `true`, BGS stops the bot when a customer leaves the chat session. |
| ChatBotHoldup | false | Optional.<br><br>If `true`, BGS enforces waiting mode for the corresponded interaction. The workflow could wait until the bot finishes the execution. For more information about bot integration with the workflow, see How chat bots integrate with the workflow. |
| ChatBotCategory | (empty string) | Optional, but strongly recommended.<br><br>Used to segregate historical reports. Max 50 characters allowed. |
| ChatBotFunction | (empty string) | Optional, but strongly recommended.<br><br>Used to segregate historical reports in additional dimension to ChatBotCategory. Max 50 characters allowed. |

Upon receiving the request, BGS checks whether this bot was already instantiated for this interaction and, if `true`, returns an error. The bot instance is identified by the pair **ChatBotID** and **ChatBotName** (where **ChatBotName** might be empty).

Otherwise, BGS:

1. Connects bot to the chat session with specified nickname and visibility level.
2. Invokes **createChatBot** methods of a bot plugin factory class in order to obtain the instance of the bot (which must implement the ChatBot interface).
3. Invokes **onCommandStart** of bot instance.

## StopBot

| Parameter | Default value | Description |
|---|---|---|
| ChatBotID | (hardcoded) | The same ID of the BGS bot provided in the ESP **StartBot** request. |
| _umsChannel | (empty string) | Required (see General notes).<br><br>Value `channel-chatbot` must be provided. |
| ChatBotName | (empty string) | Required if this parameter was initially provided in the ESP **StartBot** requests. Otherwise, |

| Parameter | Default value | Description |
|---|---|---|
| | | this parameter is optional. |

Upon receiving the request, BGS checks whether this bot was instantiated for this interaction and, if `false`, returns an error. Otherwise, BGS:

- Disconnects the bot from the chat session.
- Invokes **onCommandStop** of bot instance.
- Disposes the chat bot instance.

Important

If there is no value specified for **ChatBotId**, BGS disconnect all bots (running in the BGS instance) for a given chat session.

## CheckBot

| Parameter | Default value | Description |
|---|---|---|
| ChatBotID | (harcoded) | The same ID of the BGS bot provided in the ESP **Start** request. |
| _umsChannel | (empty string) | Required (see General notes). Value `channel-chatbot` must be provided. |
| ChatBotName | (empty string) | Required if this parameter was initially provided in the ESP **Start** request. Otherwise, this parameter is optional. |

Upon receiving the request, BGS checks whether this bot was instantiated for this interaction and, if `false`, returns an error. Otherwise, BGS returns an ESP success response.

## UpdateBot

| Parameter | Default value | Description |
|---|---|---|
| ChatBotID | (harcoded) | The same ID of the BGS bot provided in the ESP **Start** request. |
| _umsChannel | (empty string) | Required (see General notes). Value `channel-chatbot` must be |

| Parameter | Default value | Description |
|---|---|---|
| | | provided. |
| ChatBotName | (empty string) | Required if this parameter was initially provided in the ESP **Start** request. Otherwise, this parameter is optional. |
| Arbitrary list of custom parameters that will be delivered to bot as a key-value collection (at least one parameter must be present). | | |

This request allows the workflow to communicate data to the bot. The bot can communicate with the workflow by updating the userdata of the interaction (via `updateUserdata` in the `ChatBotPlatform` interface).
Upon receiving the request, BGS checks whether this bot was instantiated for this interaction and at least one custom parameter provided and, if `false`, returns an error. Otherwise, BGS invokes **onCommandUpdate** of bot instance.

## General notes

- It is strongly recommended to always specify `umsChannel=channel-chatbot` in ESP requests. This is needed in particular, when a chat session is established via any DMS channels (such as Facebook, SMS, WeChat, and so on) or when BGS is running on a DMS instance with other channels configured.
- The **Visibility** parameter defines how a bot is visible to other participants in a chat session:
  - `ALL` (conference mode) – bot is visible to all participants.
  - `INT` (coaching mode) – bot is visible only to agents and supervisors.
  - `VIP` (monitoring mode) – bot is visible only to supervisors.

### Important

A bot cannot send messages or notices to participants that do not see a bot. For example, if a bot is connected to a chat session with visibility set to **INT**, it cannot send messages to a customer like it can when visibility is set to **ALL** . When set to **INT** the bot can only send messages visible to both agents and supervisors.

# How to deploy Bot Gateway Server

## Prerequisites

- Make sure that your solution deploys the appropriate version of Chat Server (see Compatibility matrix, below).
- Deploy dedicated instance(s) of Digital Messaging Server (DMS) for Bot Gateway Server (see Compatibility matrix, below). Refer to the Digital Messaging Server 9.0.0 Deployment Procedure for more information.

### Compatibility matrix

| Component | Prerequisites |
|---|---|
| **Chat Server** | • For basic functionality, 8.5.109.06 or higher is required.<br>• To enable historical reporting, 8.5.301.06 or higher is required.<br>• To enable high availability, 8.5.308.06 or higher is required. |
| **Digital Messaging Server (DMS)** | For information on compability, see the Digital Messaging Server and Driver Compatibility page in the Genesys Interoperability Guide. |
| **Genesys Mobile Services (GMS)** | • Basic functionality (in other words, plain text messages) is supported the same way as a regular web chat.<br>• Rich Media is supported from version 8.5.201.08 and higher. |
| **Widgets** | • Basic functionality (in other words, plain text messages) is supported the same way as a regular web chat.<br>• Rich Media support must be tracked by the Widgets Release Notes.<br>• A special icon for the bot participant is supported as of version 9.0.014.09. |
| **Workspace Desktop Edition (WDE)** | • Min version 8.5.118.10 or higher is required. |

| Component | Prerequisites |
|---|---|
| | • A special icon for bot participant is supported from version 8.5.127.06 or higher.<br><br>Rich Media:<br><br>• For Web Chat—this feature is not yet supported.<br>• For Apple Business Chat—see Apple Business Chat Plug-in for Workspace Desktop Edition Release Note.<br>• For WhatsApp—see Genesys Hub Plug-in for Workspace Desktop Edition Release Note. |

## Deployment

1. Install the Bot Gateway Server either by manual installation of the IP or via the GAX deployment procedure. During the installation, you'll need to select the previously installed dedicated instance of DMS. The content of the IP will be installed in the folder `media-channel-drivers\channel-chatbot`.
2. In file "JavaServerStarter.ini" (located in the root installation folder of DMS), either locate or create two lines as shown below and set value `false` to both:
   - **-Dgenesys.mcr.stdserverex.flexchatprotocol**=`false`
   - **-Dgenesys.mcr.stdserverex.chatrequired**=`false`
3. In Interaction Server application, add DMS application to connections tab.
4. To enable high availability support, see High Availability support below.
5. Using GAX, do the following in the DMS Application object:
   a. Add connection to the following applications:
      i. Interaction Server (required for core functionality)
      ii. Universal Contact Server (required for API methods for file attachments and structured messages only).
      **Note:** If you decide not to use these methods and not connect DMS to Universal Contact Server, you must set `ucs-in-use` to `false`.
   b. In the **Application Options** tab:
      i. Remove the following sections: `[channel-any_name]`, `[channel-any_name-monitor-any_name]`, and `[endpoints:tenant_dbid]`.
      ii. Import options from the file **BotGatewayServer.cfg** (located in the **media-channel-drivers\channel-chatbot\options** subfolder in the DMS installation folder). You must de-select the check box **Overwrite existing options**. This step adds sections `[channel-chatbot]`, `[channel-chatbot-monitor-addp]`, and `[channel-chatbot-monitor-tls]`.
      iii. In section `[channel-chatbot]`, set the correct value for option `tenant-dbid` (for example, `tenant-dbid=1`).

   iv. In section `[log]`, add new option `log4j2-config-profile` with value `log4j2.xml` (if it doesn't already exist).

c. Start DMS and observe logs. The following log message indicates success:

```
Std 42111 (ChannelConnectors.configure): initialized channel-chatbot
```

**Important**

Please note that while some configuration options which are applicable to new chat session can be updated during runtime, it is generally recommended to restart DMS upon application configuration changes.

## How to uninstall or upgrade BGS

To uninstall or upgrade BGS, do the following:

1. Backup the content of the folder “media-channel-drivers\channel-chatbot\bots-repo”.
2. Stop the DMS/BGS application. This can be done via GAX (on Windows it can be done via **Services**).
3. Uninstall the BGS application:
   - On Windows through **Control Panel > Programs and Features**.
   - On Linux by running the `./uninstall.sh`, script from the "media-channel-drivers\channel-chatbot\uninstall" folder.
4. Verify that the folder “media-channel-drivers\channel-chatbot" was removed.
5. Install the new version of BGS and populate the folder “media-channel-drivers\channel-chatbot\bots-repo” from your backup.
6. Start the DMS/BGS application.

## How to test BGS

To test BGS, complete the following steps:

1. Activate the preinstalled **EchoBot**:
   a. In the **Application Options** tab of the DMS Application object, add the section `[channel-chatbot-monitor-bots]` and option `EchoBot.jar={}`.
   b. Restart the DMS application.
   **Note:** Several other bot samples are provided. You can find out how to deploy and use these other sample bots by reading the "readme.txt" file located in "media-channel-drivers\channel-chatbot\samples\demo" of the DMS installation folder.
2. Deploy the sample workflow (located in the subfolder media-channel-drivers\channel-chatbot\workflow in the DMS installation folder):

a. Import the business process:
   - either from ChatBotsGoBP.wie (require IRD and URS). Activate strategies.
   - or from ChatBotsGo.zip (requires Composer and ORS). Run **Generate All** (ignore the error against "RESTProxy.jsp").

b. If you have more than one DMS instance running (and configured in connections of Interaction Server application), you need to select your Bot Gateway Server application in the **External Service** block which starts the bot (located in **Start** strategy).

3. Launch the Chat Widget with the following userdata:

```
{ ChatBotID: "EchoBot"
 , ChatBotName: "EchoBot"
 , ChatBotHoldup: "false"
 , StopBotOnAgentArrival: "false"
 , StopBotOnCustomerLeft : "false"
 , Visibility : "ALL"
 , Nickname : "Demo Echo Bot"
 , _umsChannel : "channel-chatbot"
 }
```

4. As soon as the chat session starts in the Chat Widget, EchoBot connects and it echos every posted message into the chat session. You can stop EchoBot by sending one of the following texts in the chat session:

```
stop:keep_alive
 stop:force_close
 stop:close_if_no_agents
 stop
```

**Note:** In waiting mode (in other words, when `ChatBotHoldup=true`), with default configuration of `holdup-attribute-name` (which is “ScheduledAt”) Bot Gateway Server relies on the scheduled interactions functionality of Interaction Server. If this conflicts with your workflow logic, you can set `holdup-attribute-name` to another value. In this case you will need:

- To modify Interaction Server database schema and create correspondeing interaction custom property of type “timestamp” (refer to Custom Properties in the eServices Interaction Properties Reference Manual).
- To specify view condition in “Waiting” queue instead of using “Scheduling” (which then must be set to “Ignore Scheduling”).

## High availability and sizing guidelines

### High Availability support

High availability (HA) support consists of the following functional areas:

| Area | Deployment Notes |
|---|---|
| Bot restoration after BGS switchover | The following must be done: |

| Area | Deployment Notes |
|---|---|
| | 1. Add a special custom interaction attribute which is used by BGS for the interactions' content query in HA mode. For that:<br>a. Stop Interaction Server.<br>b. Upgrade the Interaction Server Database (execute script `ha_support_<database name>.sql` from the **ha_sql_scripts** folder).<br>c. Create a new interaction custom property. In **Configuration** (using GAX):<br>i. Navigate to the **Business Attributes** folder and find (or create) a new **Business Attribute** with:<br>• **type**: `Custom`<br>• **name**: `InteractionCustomProperties`<br>• **display name**: `Interaction Custom Properties`<br>ii. Navigate to the **Attribute Values** of `InteractionCustomProperties` and create a new value with the **name** and **display name** of `GCTI_BGS_AppName` and a `[translation]` section in the **Annex** tab with the option **translate-to** and value of `bgs_app_name`.<br>d. Start Interaction Server.<br>2. Create two BGS applications and configure those as the primary/backup pair.<br>3. For each application (primary and backup), set the configuration option ha-bot-recovery-enabled with the value of `true`. Optionally, revise if you need to adjust the values for options ha-bot-start-unrestricted and ha-bot-recovery-batch. |
| Bot context saving and restoring | Ensure that Chat Server (minimum version 8.5.308.06) is deployed in HA mode so the bot context will be preserved as a part of the chat session transcript in UCS or Cassandra. |
| Reconnection to chat session | BGS supports the automatic (seamless) reconnection to the chat session in the case of a Chat Server failure. Revise the chat session reconnect schedule defined in the configuration options ha-chat-reconnecting-attempts and ha-chat-reconnecting-delay. Genesys recommends deploying HA for Chat Server in N+1 (load-balancing) mode, along with adjusting a few |

| Area | Deployment Notes |
|---|---|
| | configuration options as described in the topic, Configure Chat Server for HA. |
| Reconnection to Interaction Server and UCS | Reconnection to Interaction Server and UCS is defined via a standard application connection configuration.<br><br>**Note:** UCS connection is needed only for Rich Media and attachments functionality. In order for the reconnection to Interaction Server and UCS to be enabled, these applications must be configured as a primary/backup pair |

## Sizing guidelines

For High Availability (HA) support, no special configuration on BGS side is required, unless you need to define a different chat session reconnect schedule with configuration options ha-chat-reconnecting-attempts and ha-chat-reconnecting-delay. BGS supports the automatic (seamless) reconnection to the chat session in case of a Chat Server failure, and it also handles the reconnection to Interaction Server and UCS (the later is needed only for Rich Media and attachments functionality). In order for the reconnection to Interaction Server and UCS to be enabled, these applications must be configured with a primary/backup pair. Genesys recommends deploying HA for Chat Server in N+1 (load-balancing) mode, along with adjusting a few configuration options as described in the topic, Configure Chat Server for HA.

For BGS, the following performance benchmarks were obtained on hardware with "Intel Xeon E7-8880L 2 GHz" and a single instance of Bot Gateway Server. The average length of a chat session was around 55 seconds (with 5 short messages from a customer and 5 short messages from an agent), which is a quite dense scenario. A single instance of BGS was capable of maintaining around 2500 concurrent bot sessions while consuming about 2 CPU cores and 3 GB of RAM. With this given scenario, on-average about 50 new bots started every second. hen planning the deployment, the chat session density (in other words the number of messages from or to a bot, each second) and the bot startup rate (how many new bots start, each second) play a significant role in the BGS performance benchmarking.

## Configure secure and ADDP connections

You can configure the following secure and ADDP connections:

| Connection | Configuration |
|---|---|
| • DMS to Configuration Server<br>• DMS to Message Server<br>• DMS to Interaction Server<br>• DMS to Universal Contact Server<br>• Interaction Server to DMS | Use standard Genesys approach to provide TLS configuration on hosts, applications, ports, and ADDP parameters in connections |

| Connection | Configuration |
|---|---|
| DMS/BGS to Chat Server | Use a special approach described below. |

### Secure connection to Chat server

When connecting a bot instance to a chat session, Bot Gateway Server connects to the host/port which is specified in the userdata of the interaction as `ChatServerHost`/`ChatServerPort`. If a port is configured as secure, Chat Server also attaches `ChatServerConnType=tls`. When Bot Gateway Server detects this property in the userdata, it tries to establish a secure connection. To configure these secure connections, you need to use section `[channel-chatbot-monitor-tls]` of the DMS application:

1. Set option `provider` to a desired security provider (see description of options for more information). By default, a provider is set to `ANY`, which will allow you to establish a secure connection without using any certificate. This mode is recommended for testing purposes only.
2. Configure option `trusted-ca` with corresponding certificate.
3. Review and configure (if needed) all other options in this section.
   **Note:**
   - Mutual TLS mode is not supported for BGS connection to Chat Server.
   - BGS uses Java PSDK to communicate with Chat Server, for more information please refer to TLS and the Platform SDK Commons Library.

### ADDP connection to Chat server

Use section `[channel-chatbot-monitor-addp]` in order to configure ADDP connection to Chat Server.

## Enable Reporting for BGS

Genesys Historical Reporting Solution provides reports about chat bots that are run by BGS. The reporting solution consumes reporting data from BGS via Kafka integration. For full information, see Integrating BGS with Genesys Historical Reporting.

## GAX deployment

The following steps describe how to use Genesys Administrator Extension (GAX) to deploy the Bot Gateway Server.

1. **Upload the driver installation package and template:**
    a. Log in to GAX.
    b. Go to **Administration > Installation Packages**.
    c. Click **+** (New) to upload a new installation package.
    d. In the **Software Installation Wizard** panel, select **Installation Package Upload (includes templates)**.
    e. Click **Next**.
    f. The panel updates. Click **Choose File** to select the file to upload.
    g. Click **Finish**.
    h. The file begins uploading from your file system to GAX. When the upload is complete, the IP is displayed in the **Installation Packages** list.
2. **Deploy the driver installation package and template:**
    a. In GAX, go to **Administration > Installation Packages**.
    b. Click the **Bot Gateway Server** installation package to view its properties.
    c. Verify the Status field has a value of **Complete**.
    d. Click the gear icon and select **Install** to open the IP Deployment Wizard panel.
    e. Click **Next**.
    f. In the **Host** section check needed host.
    g. Click **Next**.
    h. In the **Application Parameters** pass correct Digital Message Server application name to **Application name for host**.
    i. Select correct **Primary Configuration Server** and **Backup Configuration Server**.
    j. Click **Next**.
    k. In the **Installation Parameters** section:
        - For Linux: set path **DMS folder+/media-channel-drivers/channel-chatbot** to **IPCommon: InstallPath**.
        - For Windows: set path **DMS folder** to **IPCommon: PathToDigitMsgSrv**.
    l. Click **Next**.
    m. In the **Deployment** section click **Finish**.
    n. The BGS begins deploying to the selected host under the selected DMS.

o. When the deployment is complete, click **Close**.

# Integrating BGS with Genesys Historical Reporting

For Bot Gateway Server (BGS), historical reporting on chat bot activity supplements the chat session reporting available in eServices premise deployments that include the Genesys Reporting & Analytics offering. Chat bot reporting is supported only for chat bots that are run by BGS.

This page describes the component and configuration requirements to enable historical reporting on BGS-managed chat bot activity in your deployment.

## Overview: BGS reporting process

1. After a BGS session is finished or when an attempt to launch a bot session is rejected, BGS produces a reporting event, which it publishes to Kafka. For more information about the reporting event attributes, see Reporting event attributes, below.
2. The BGS reporting event is separate from the Interaction Server reporting event that is generated at the end of the Chat Server session. The chat session reporting event includes some bot-related statistics that are processed as part of chat session reporting (see Integrating Chat Server with Genesys Historical Reporting in the *Chat Server Administrator's Guide*).
3. On a regular schedule, Genesys Info Mart extracts the BGS data from Kafka and transforms it into the BGS_SESSION_FACT table and supporting dimensions in the Info Mart dimensional model. For more information about the Info Mart database tables, see the *Genesys Info Mart Physical Data Model* for your RDBMS. For more information about managing the Genesys Info Mart ETL jobs, see the *Genesys Info Mart Operations Guide*.
4. In deployments that include Reporting and Analytics Aggregates (RAA) and Genesys CX Insights (GCXI), RAA summarizes and organizes the Info Mart data in ways that enable GCXI to extract meaning. For more information about RAA data, see the *RAA User's Guide*.
5. GCXI uses the aggregated data in the Info Mart database to produce a Bot Dashboard. For more information, see Chat reports in the GCXI *User's Guide.*

## Enabling historical reporting on BGS activity

### Prerequisites

The following table summarizes the minimum release requirements for the Genesys and third-party components that enable chat bot historical reporting.

| Component | Minimum release |
|---|---|
| Bot Gateway Server | 9.0.004 |
| Kafka | 2.0 |

| Component | Minimum release |
|---|---|
| Chat Server | 8.5.203.09 |
| Genesys Info Mart | 8.5.011.18 |
| RAA | 8.5.003 |
| GCXI | 9.0.005 |

## Setting up historical reporting

1. **Ensure that your deployment has been configured as required for Genesys Info Mart to support chat session reporting.**
   For more information, see Integrating Chat Server with Genesys Historical Reporting in the *Chat Server Administrator's Guide.* If you have not already done so, configure Interaction Concentrator (ICON) to store the user data KVPs listed below (see Chat Server reporting data).
2. **Configure BGS to report bot metrics.**

   By default, BGS captures the minimum attributes required in the reporting event to enable historical reporting out-of-box. However, the default ESP methods do not populate all the parameters that are useful for reports.

   For meaningful reporting, Genesys strongly recommends that you populate the **chatBot_category** and **chatBot_function** attributes, in particular. There are two ways you can populate the category and function attributes:

   - Through the API (in **BotCreationAttributes**) during **createChatBot**
   - By specifying **ChatBotCategory** and **ChatBotFunction** in the parameters of the ESP **StartBot** method

   For more information about the available attributes, see ESP methods and BGS reporting event attributes, below.
3. **Enable the storage of BGS reporting metrics in Kafka.**
   a. Deploy Kafka version 2.0.
   b. Configure BGS to output reporting data into Kafka by configuring the following options in the **channel-chatbot** configuration section:
      - kafka-cluster-brokers or kafka-zookeeper-nodes
      - kafka-topic-name. The default value is `chat-bots-reporting`.
4. **Configure Genesys Info Mart to extract the BGS reporting data from Kafka.**
   a. On the **Options** tab of the Genesys Info Mart application object, create a new configuration section, **kafka-<cluster-name>**. The <cluster-name> can be any string you use to identify the cluster—for example, `kafka-1`.
   b. In the new section, add the following options:
      - bootstrap.servers—The value must match the value of the BGS **kafka-cluster-brokers** or **kafka-zookeeper-nodes** option (see step 3).
      - g:topic:<topic-name>—The <topic-name> must match the value of the BGS **kafka-topic-name** option—for example, **g:topic:chat-bots-reporting**. The value of the option must be BGS_K.
   c. (Optional, but recommended) Set an alarm on log message 55-20049, which identifies that a transformation job error has occurred because of a Kafka exception, such as a complete loss of connection to the cluster.

5. **Enable aggregation of bot-related data. (Required for GCXI reporting or other applications that use RAA aggregation.)**
   In the **[agg-feature]** section on the Genesys Info Mart application object, specify the enable-bgs option. If you haven't already done so, also specify the enable-chat option.

## Bot-related reporting data

There are two mechanisms by which Genesys Info Mart receives bot-related reporting data:

- BGS application data
- Chat Server reporting data

### BGS application data

After a BGS session is terminated or rejected, BGS generates a reporting event for that session and stores the data in Kafka. There might be multiple BGS sessions within a single chat session.

#### JSON Example

The following is an example of a BGS reporting event serialized as a JSON file for Kafka storage. See BGS reporting event attributes for the meaning of the attributes.

Click to see example.

#### BGS reporting event attributes

The following table describes the attributes included in the BGS reporting event. The "Application data attribute" column, which includes the name of the section as well as the attribute itself, represents the XPath search term Genesys Info Mart uses to extract and map the data. The "Info Mart Database Target" column indicates the Info Mart database table and column to which the attribute is mapped.

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| /cbs_endTime | The UTC-equivalent value of the date and time at which the BGS session ended or was rejected. | BGS_SESSION_FACT.END_TS, BGS_SESSION_FACT.END_DATE_TIME_KEY |
| /chatBot_info/chatBot_category | The generic category describing the type of function performed by the bot, such as Monitoring, Dialog, Notification, or Service. **Default value:** "Unspecified" | BGS_BOT_DIM.BOT_CATEGORY (referenced through BGS_SESSION_FACT.BGS_BOT_DIM_KEY) |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| /chatBot_info/chatBot_function | The specific bot functionality, such as Translator, Advisor, Escalation, Recording, AI, or Questioner.<br>**Default value:** "Unspecified" | BGS_BOT_DIM.BOT_FUNCTION (referenced through BGS_SESSION_FACT.BGS_BOT_DIM_KEY) |
| /chatBot_info/chatBot_name | The identification of the bot represented by "ChatBotID-ChatBotName" pair, where:<br>• **ChatBotID (always presented)** - The ID of the BGS bot plugin (this ID is hardcoded inside the bot and returned by `getBotId()`).<br>• **ChatBotName (may be empty)** - The name of the "external" bot (for example, if the bot plugin implements a connector to other bot frameworks). | BGS_BOT_NAME_DIM.BOT_NAME (referenced through BGS_SESSION_FACT.BGS_BOT_NAME_DIM_KEY) |
| /session_info/attr_itx_id | The interaction GUID, as reported by Interaction Server. This value is the ID of the chat session. | BGS_SESSION_FACT.MEDIA_SERVER_IXN_GUID |
| /session_info/attr_itx_media_type | The media type of the parent interaction. The default value of "NONE" means that BGS was unable to get information about the interaction.<br>**Valid values:** [Chat, Email, sms]<br>**Default value:** "NONE" | MEDIA_TYPE.MEDIA_NAME_CODE (referenced through BGS_SESSION_FACT.MEDIA_TYPE_KEY) |
| /session_info/attr_itx_submitted_at | The timestamp of the start of the interaction (in other words, the chat session) in Interaction Server. The default value is used when BGS is unable to get information about the interaction.<br>**Default value:** Current DateTime in ISO8601 date format as provided by Interaction Server | BGS_SESSION_FACT.INTERACTION_SDT_KEY |
| /session_info/attr_itx_tenant_id | The DBID of the Tenant. The default value is used when BGS is unable to get information about the Interaction.<br>**Default value:** -1 | BGS_SESSION_FACT.TENANT_KEY |
| /session_info/cbs_id | The ID assigned by BGS to every bot instance or process connected to the chat session. | BGS_SESSION_FACT.CBS_ID |
| /session_info/cbs_rejectedToStart | Flags whether the session was | BGS_SESSION_DIM.REJECTED_TO_START |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| | rejected before it started. If the session was rejected (the value of the attribute is 1), the "session_stats" section of the reporting event is omitted.<br>**Valid values:** 0, 1 | (referenced through BGS_SESSION_FACT.BGS_SESSION_DIM_KEY) |
| /session_info/cbs_startTime | The UTC-equivalent value of the date and time at which the bot session was initiated in BGS, regardless of whether the session was accepted or rejected. | BGS_SESSION_FACT.START_TS, BGS_SESSION_FACT.START_DATE_TIME_KEY |
| /session_stats/cbs_duration | The duration, in milliseconds, of the BGS session. | BGS_SESSION_FACT.DURATION |
| /session_stats/ cbs_endedAbnormally | Indicates whether the session ended abnormally for a technical reason (for example, a protocol or connection error resulted in disconnection of the bot from the session)<br>**Valid values:** 0, 1 | BGS_SESSION_DIM.ENDED_ABNORMALLY (referenced through BGS_SESSION_FACT.BGS_SESSION_DIM_KEY) |
| /session_stats/cbs_endedBy | The type of participant that initiated termination of the BGS session.<br>**Valid values:**<br>• SYSTEM—Denotes a Chat Server (for example, if Chat Server ended the chat session because of idle control or shutdown).<br>• AGENT—Denotes an agent who participated in the chat session, regardless of whether that agent was visible to the customer.<br>• BOT—Denotes the chat bot.<br>• CBP—Denotes Bot Gateway Server (for example, if BGS ended the session because all participants left, or as a reaction to `StopBotOnCustomerLeft` if provided during bot startup, or during server shutdown, or because of a malfunction). | BGS_SESSION_DIM.ENDED_BY (referenced through BGS_SESSION_FACT.BGS_SESSION_DIM_KEY) |
| /session_stats/cbs_endReason | The reason the BGS session was terminated.<br>**Valid values:** | BGS_SESSION_DIM.END_REASON (referenced through BGS_SESSION_FACT.BGS_SESSION_DIM_KEY) |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| | • ESP_REQUEST — Stopped by ESP request.<br>**Possible associated values for:**<br>• cbs_endedBy: SYSTEM<br>• cbs_endedAbnormally: 1/0<br>• AGENT_JOINED — Configured with **StopBotOnAgentArrival**, and agent joined.<br>**Possible associated values for:**<br>• cbs_endedBy: CBP<br>• cbs_endedAbnormally: 1/0<br>• ALL_CLIENTS_LEFT — Configured with **StopBotOnCustomerLeft**, and the last client (cusomer) left.<br>**Possible associated values for:**<br>• cbs_endedBy: CBP<br>• cbs_endedAbnormally: 1/0<br>• ALL_PARTICIPANTS_LEFT — All participants (except bots, external users, and system users) left the session.<br>**Possible associated values for:**<br>• cbs_endedBy: CBP<br>• cbs_endedAbnormally: 1/0<br>• RELEASED_BY_PARTY — A party release event was received, and the asker was not the current bot.<br>**Possible associated values for:**<br>• cbs_endedBy: AGENT or BOT<br>• cbs_endedAbnormally: 1/0<br>• RELEASED_BY_SELF — A party release event was sent by the bot (through **leaveSession**).<br>**Possible associated** | |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| | **values for:**<br>• cbs_endedBy: BOT<br>• cbs_endedAbnormally: 1/0<br>• REMOVED_BY_SERVER — Chat Server removed a bot participant.<br>**Possible associated values for:**<br>• cbs_endedBy: SYSTEM<br>• cbs_endedAbnormally: 1/0<br>• IDLE_CONTROL — A bot participant was removed because the chat session was closed because of inactivity.<br>**Possible associated values for:**<br>• cbs_endedBy: SYSTEM<br>• cbs_endedAbnormally: 1/0<br>• DISCONNECTED — A protocol or connection error resulted in disconnection from the session.<br>**Possible associated values for:**<br>• cbs_endedBy: SYSTEM/CBP<br>• cbs_endedAbnormally: 1<br>• CBP_SHUTDOWN — CBP was shutting down.<br>**Possible associated values for:**<br>• cbs_endedBy: CBP<br>• cbs_endedAbnormally: 1/0 | |
| /session_stats/cbs_endResult | Not currently populated by BGS. In the future, this attribute might be populated with information provided by bots about the business result of the session: Success or Fail. | BGS_SESSION_DIM.END_RESULT (referenced through BGS_SESSION_FACT.BGS_SESSION_DIM_KEY) |
| /session_stats/ cbs_messagesReceived | The number of messages received by the bot in the BGS session. | BGS_SESSION_FACT.MESSAGES_RECEIVED |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

| Application data attribute | Description | Info Mart Database Target |
|---|---|---|
| /session_stats/cbs_messagesSent | The number of messages sent by the bot in the BGS session. | BGS_SESSION_FACT.MESSAGES_SENT |
| **Application data attribute** | **Description** | **Info Mart Database Target** |

## Chat Server reporting data

When the chat session is finished, Chat Server attaches reporting statistics to the user data of the interaction in Interaction Server.

The following table describes the bot-related reporting statistics that Chat Server includes in the user data if any BGS-managed chat bots participated in the chat session. The "Info Mart Database Target" column indicates the Info Mart database table and column to which the user data KVP is mapped. (For information about the rest of the chat session KVPs that Chat Server sends, see Chat Server reporting data in the *Chat Server Administration Guide.*)

### Important

If the BGS session is rejected, no KVPs related to the rejected session are included in the user data that Chat Server attaches, regardless of the attach-session-statistics option value.

| KVP | Description | Info Mart Database Target |
|---|---|---|
| csg_MessagesFromBotsCount | The total number of messages visible to the customer that were sent by all bots that participated in the chat session. | CHAT_SESSION_FACT.MSG_FROM_BOTS_COUNT |
| csg_MessagesFromBotsSize | The total character count (including spaces) of all messages sent by bots that participated in the chat session. | CHAT_SESSION_FACT.MSG_FROM_BOTS_SIZE |
| csg_PartiesAsBotCount | The number of parties that participated in a chat session as bots. If the same bot (in other words, a bot with the same ID) connects multiple times, it is counted as a separate participant each time it joins. | CHAT_SESSION_FACT.BOTS_COUNT |
| csg_SessionUntilFirstBotTime | The duration of the waiting period, or the period of time a customer waits until the first bot (visible to a customer) joined the chat session. | CHAT_SESSION_FACT.UNTIL_FIRST_BOT_DURATIO |
| **KVP** | **Description** | **Info Mart Database Target** |

| KVP | Description | Info Mart Database Target |
|---|---|---|
| | **Note:** The 0 (zero) value has two alternative interpretations: No bots ever joined the session (if csg_PartiesAsBotCount=0) or a bot joined immediately when the chat session was started (if csg_PartiesAsBotCount > 0). | |
| **KVP** | **Description** | **Info Mart Database Target** |

# How to develop a new bot for Bot Gateway Server

## Prerequisites

1. Install Java Development Kit 1.8.
2. Install NetBeans 8.2 (https://netbeans.org/downloads/). You can use the edition listed as **Java SE** (alternatively, you can use any other Java IDE.)
3. Install "Apache Maven 3". You can either:
    - Install it from https://maven.apache.org/download.cgi.
    - Install a Java IDE with Maven embedded.
    - Reuse embedded Maven in NetBeans (in Windows the **mvn** application is located in the NetBeans installation folder under **java\maven\bin**).

## Preparing the environment for a new bot project

1. Obtain Bot Gateway Server (BGS) project template files (located in the subfolder **media-channel-drivers\channel-chatbot\provision** in the DMS installation folder).
2. Import the files into a local Maven repository by executing the following commands from the location where the project template files are stored:

```
> mvn org.apache.maven.plugins:maven-install-plugin:2.5.2:install-file
 -Dfile=ChatBotArchetype.jar -DpomFile=pom.xml

> mvn org.apache.maven.plugins:maven-install-plugin:2.5.2:install-file
 -Dfile=ChatBotApi.jar -Djavadoc=ChatBotApi-javadoc.jar
```

### Important

You must either add the location of the **mvn** application into PATH or use the full-path specification when running the commands.

3. Update the archetype catalog in your local Maven repository with the following command:

```
> mvn archetype:update-local-catalog
```

**Important**

If the above command failed or if the archetype catalog was not created yet, run the following command:

```
> mvn archetype:crawl
```

4. Restart NetBeans (if NetBeans is running).

## Creating a new bot project

1. Using NetBeans IDE, do the following:
    a. From the **File** menu select **New Project**.
    b. Select **Maven > Project from Archetype**.
    c. Click **Next**.
    d. Use the **Search** field to find the **ChatBotArchetype** archetype or select from the **Known Archetypes** list.
    e. Click **Next**.
    f. Modify the following:
        - **Project Name** - Name of the project (same as artifact ID)
        - **Project Location** - Where project will be placed
        - **Group Id** - Maven groupId
        - **Version** - Project version
        - **Package** (Optional) - Package for Java classes (should be generated based on **Group Id** and **Project Name**)
    g. Click **Finish**.
2. Implement your bot by following the directories from Bot implementation guidelines
3. Build the project:
    1. Right-click on the project from the **Projects** view.
    2. Select **Clean and Build**. The resulting JAR file is in the **target** subfolder of the project.

## Deploy a new bot

1. Copy the bot plugin JAR file to the subfolder **media-channel-drivers/channel-chatbot/bots-repo** in the DMS installation folder.
2. Enable the bot plugin. Open the DMS Application object and go to the section `[channel-chatbot-`

`monitor-bots]` (you must create the section if it is absent). Add an option where:

- The key contains the name of the bot plugin jar file (which is placed into the **bots-repo** folder). For example: `MySampleBot.jar`.
- The value provides the configuration for the bot, which is provided to the bot with method **initialize** of type **KeyValueMap**. This value can:
    - Contain {} (an empty configuration)
    - Contain a JSON string (enclosed between {} without line breaks)
    - Contain the full path to the file with JSON (line breaks allowed). The file name can be absolute or relative (in case of relative filename, the **bots-repo** directory is treated as a root folder). Use this approach if you need to provide sensitive data (for example, password) in bot configuration.

3. Observe the DMS logs to ensure the bot loads successfully.
4. Provide a workflow which starts, stops, and manages the bot.

Important

- If you want to disable the bot: Remove the corresponding option from the section `[channel-chatbot-monitor-bots]`.
- If you need to update the configuration file for the bot:
    - Preserve the original file by copying the JSON configuration file (a path to this file is specified in the DMS Appliction object) and renaming the copied file.
    - Update the file as needed, then save your changes.
    - Update the the corresponding option in section `[channel-chatbot-monitor-bots]` with the new filename.
- If you need to replace the bot plugin JAR file, you must stop DMS, replace the JAR file, and then restart DMS.

## Bot implementation guidelines

### Bot implements two interfaces

- BGS invites bots into chat session via ChatBotFactory interface and then relays the chat session activity to the bot via the ChatBot interface. BGS also initializes and provides bot configuration through the ChatBotFactory interface. You must implement the bot's logic in the methods of the ChatBot interface.
    - You can modify and/or extend the default implementation of the ChatBotFactory interface. For example, you can change the `getBotId()` method if a different bot ID is needed.
    - When implementing methods of the ChatBot interface, you must use the ChatBotPlatform `cbpInstance` class in order to send messages or notices into the chat session, leave the chat

session, and update the userdata of interaction in workflow.

    - If you need to change the name and/or package of the ChatBotFactory implementation class, update the file "com.genesyslab.chat.bots.chatbotapi.ChatBotFactory" in folder "src\main\resources\META-INF\services" of your project.
    - The implementation of `ChatBot` and `ChatBotFactory` interface must not introduce new methods with the same names (and different parameters) which are already defined in the interface.
    - It is recommended to use `ESP_REQUEST` as a source in `getUserData(UserDataSource)` of `getInteractionInfo()` in `GenesysChatSession` to avoid a possible race condition between workflow and bot launching.

- The bot must use the logger obtained through "com.genesyslab.chat.bots.chatbotapi.platform.LoggerFactory" or `ChatBotPlatform.getLogger()`. The bot must not use the "slf4j" logger directly.

**Important**

Since a single thread is being used to deliver all notifications to a bot, the methods implemented in the `ChatBot` and `ChatBotFactory` interfaces must not delay the thread which calls those methods, . Also, the esp-proc-timeout configuration option is what specifies the number of seconds to process all incoming ESP requests from the workflow.

## Bot uses interfaces provided by BGS

This functionality can be described as:

- Core functionality provided by ChatBotPlatform interface which allows a bot to send messages and notices to other participants, leave chat session, communicate to workflow (`updateUserdata`) and provide access to other interfaces. For guidance, you can check the implementation of basic bot functionality in EchoBot (the project files are located in the subfolder "media-channel-drivers\channel-chatbot\samples" of the DMS installation folder).
- File attachment API allowed to decipher notice events from other participants into file attachments (if notice represents it), to send a file attachment which can be created from raw file or from a file attached to a standard response. BGS IP contains `BalloonsBot` and `ThumbnailsBot` which demonstrate how to use the API for file attachments.
- Structured messages (also known as "rich" messages) API permits rich messages to be sent in a channel where those are supported and to process replies. BGS IP contains CafeBot which demonstrates the capabilities of this API, including the use of Standard Reponses for rich messages. Through the use of rich messages, the BGS API provides the `getRelatedEventId()` method which defines correspondence between sent structured messages and a received reply.

## High availability mode recommendations

By default, high availability mode does not require any special handling unless the bot needs to process special situations (these features are demonstrated in a new BobberBot sample project). In this case, the bot can:

- Use **HighAvailabilityManager** methods to save the bot context and to retrieve it upon bot restoration.

- Implement a **ChatDisconnectHandler** interface to process a disconnection and reconnection to a chat session.

**Notes:**

- BGS is a hosting environment for a bot JAR file which is simply a Java application. You can communicate with any 3rd party services (via REST HTTP API) from within the bot implementation, the same way as you can from any ordinary Java application.
- To implement agent escalations, you'll need to determine how the bot communicates with the workflow (in other words, URS/ORS strategies) which handles the corresponding interactions. For example, you can force the workflow to wait on a value condition of a certain userdata key/value pair (KVP), and then update this KVP from the bot when it needs to escalate to an agent. Or, alternatively, you can run the bot in so-called ”waiting” mode (see the sample workflow) and simply finish the bot execution (call `leaveSession()`) when it needs to escalate to an agent, so the workflow continues the routing.

## Using 3rd party libraries

- If your bot needs to use 3rd party libraries, you have the option to use `maven-assembly-plugin` and these libraries will be included into the resulting JAR file.
- Each bot is launched by a separately instantiated class loader to avoid issues such as unfulfilled JAR file dependencies or JAR version inconsistencies.