Genesys Knowledge Center API Reference

Overview

5/2/2026

Overview

The Genesys Knowledge Center REST API exposes three sets of functionality:

- The Knowledge API can be used by Knowledge Center Server clients who are interested in retrieving FAQ-related information from a knowledge base, including things like the structure of the knowledge base and its feedback data
- The Management API allows service components—such as content management systems, the Knowledge Center Administrator plugin, and data importers—to create, populate, and manage knowledge bases
- The Reporting API provides reporting engines—such as Easy Pulse or third-party products—with data on the various knowledge-related activities carried out by agents and customers

Structure

- The methods in this API follow the structure and meaning documented below unless the method description explicitly states otherwise.
- The default data format is JSON.

Structures

- **/{resource-type}**—URIs that have this format represent a set of instances of the specified resource type and are plural (for example, **kbs** or **categories**), unless the resource is a singleton in the specified context. Here are the default meanings of these operations:
  - **GET**—Retrieves a list of all instances of the resource
  - **POST**—Creates a new instance of the resource
  - **PUT**—Updates a set of instances of the resource
  - **DELETE**—Deletes a set of instances of the resource.
- **/{resource-type}/{id}**—URIs that have this format represent a specific instance of the resource type. Here are the default meanings of these operations:
  - **GET**—Retrieves information about the instance.
  - **POST**—N/A
  - **PUT**—Updates the instance with the specified information.
  - **DELETE**—Removes the instance.
- **/{resource-type}/{id}/{operation}**—URIs that have this format represent a non-CRUD operation for a specific instance of of the resource type. Here are the default meanings of these operations:
  - **GET**—The executed operation is intended to retrieve data without modifying the underlying data

- **POST**—The executed operation is intended to update or add data to the underlying data
- **PUT**—N/A
- **DELETE**—N/A

Versioning

- If a previous version of the API is to be used, resource URI's must contain a prefix of **/v*{ordinal}*/** to identify that fact.
- If no prefix is specified, the URI refers to the most recent version of the API.

Client identification

For identifying client and related to API call mediatype, you must provide the values of two specific header variables at each API call. Any persistence (session,cookies) for these variables is not provided.

Identifies	Header Variable Name
Client application	gkc_apiClientId
Client media type	gkc_apiClientMediaType

Authorization

You can use these APIs to access public knowledge bases without requiring authorization, whether on the part of your agents or your customers. However, for all other access, including agent-related queries against internal knowledge bases, you must provide the appropriate agent credentials.

The required authorization levels are described in the individual API reference sections.

Authentication Overview

Genesys Knowledge Center Server doesn't have its own authentication mechanism. It is the responsibility of the broader Genesys environment or your corporate site to ascertain that somebody really is who he or she claims to be. This is applicable both for customers and agents. Applications that post requests to Knowledge Center Server will need to notify Knowledge Center Server of the validated identity of customers and agents.

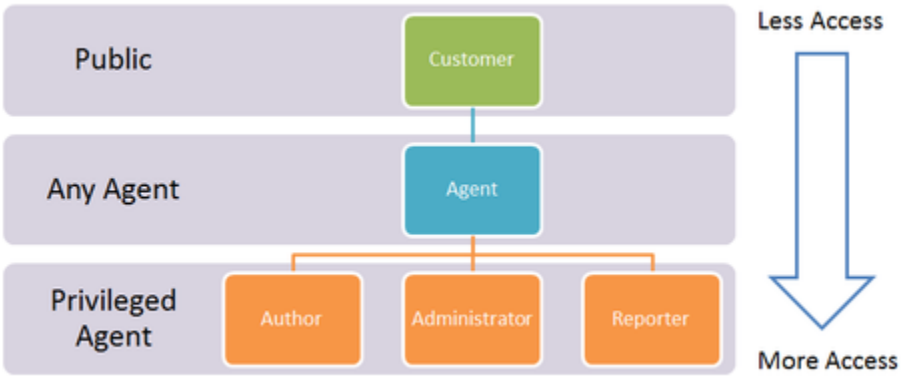
Authorization

There are two principal types of users:

- Customers

- Agents

Customers are allowed to access public information only. They do not have any extra permissions or access rights. Unless it is proven that a user is an agent, customer-level permissions will be used.



Access Roles for the REST APIs

Because of this, users who have a *Customer* role do not require authorization.

Agent is the most basic role in the contact center. The roles of *Administrator*, *Author*, and *Reporter* have additional capabilities it that can be granted using privileges in the Genesys Configuration Layer.

When subtenancy functionality is enabled, agents can only access data from within their own subtenant. Also note that an agent can only belong to one subtenant at a time.

Available http-header variables

There set of http header variables are used by Genesys Knowledge Server for identifying various aspects of API call environment. These header variables are: Table:

Name	Type	Description
gkc_agentId	String	Unique identifier of Genesys agent that interacts with server through a agent UI .
gkc_customerId	String	Unique identifier of external customer.
gkc_interactionId	String	Unique identifier of interaction in Genesys environment. This header is mandatory for call of PUT /feedback/interaction (registering incoming interaction into Genesys Knowledge history).

Authorization, Agent and Customer Credentials

In order to get the appropriate level of authority, Knowledge Center Server receives agent and customer identifiers using special request header variables:

Identifies	Header Variable Name
Agent	gkc_agentId
Customer	gkc_customerId

Wherein **gkc_agentId** in the request that is passed to the Genesys Knowledge Center Server must correspond to the identifier of the user configured Genesys Configuration Server. Genesys Knowledge Center server uses the value of **gkc_agentId** for authorizing the call by credentials of the identified user that is configured in the Genesys Configuration Server.

The **gkc_customerId** request header parameter does not affect any restrictions on request processing, but is used as an identifier for collecting the history log in relation to a particular customer.

Examples

```
POST /v1/kbs/search HTTP/1.1
gkc_customerId: some_customer_id
Content-Type: application/json
...

POST /v1/kbs/sample/langs/en/docs/delete HTTP/1.1
gkc_agentId: genesys_agent_id
Content-Type: application/json
...
```

> **Important**
>
> It is possible to save the customer identifier into the Genesys Knowledge Center session storage. In this case you must use strict session-oriented GKC protocol.

Start

1. Associate customer id into session and obtain the session ID that is related to this customer ID:

   ```
   PUT /v1/session/new
    gkc_customerId: some_customer_id
    Content-Type: application/json
    ...
   ```

   If the operation is successful, you'll receive a response with this structure:

Field	Type	Description
statusCode	String	"OK" in case of success
error	String	null on success or error

Field	Type	Description
		description
response	SessionInfo	Response payload
sessionId	String	Session identifier

2. Now you can use the sessionId resulting from from Step 1 as a parameter to the subsequent REST API calls to Knowledge Center server, as shown here:

```
POST /v1/kbs/search?sessionId=<obtained-session-id> HTTP/1.1
gkc_customerId: some_customer_id
Content-Type: application/json
{
    ...
}
```

End

Request and Response Descriptions

Request Execution Handling

- Each response contains an execution code along with the details of the response
- The execution codes have the following meanings:

HTTP Code	HTTP Description	Usage Guidelines
200	OK	The method has executed successfully
400	Bad Request	The request cannot be executed, as it is missing one or more mandatory parameters (parameters are missing in the query string or the request body)
401	Unauthorized	The user does not have permission to access this method
404	Not Found	The resource type or resource ID specified in the URI is unknown
500	Internal Server Error	The request is valid, but the server had trouble executing it
503	Service Unavailable	The server was unable to serve the request at the time it was processed; the request may be resent later

- Execution codes are returned both as HTTP Response codes and in the special field of the response body (in JSON format)
- All response bodies returned by the server have the following standard high level structure:

```
{
    "statusCode": <execution code>,
        "error": {
            "type": <string>,
            "message": <string>
        },
```

```
        "response": {
            ...
        }
}
```

Responses

Field	Type	Required	Description
statusCode	RestStatus	Yes	Execution code of the operation
error	Error	Yes	Detailed information on the error message. Present only if the operation executed was unsuccessfully.
response	complex	No	Response for the executed operation. This can be missing if an error occurred or if the method has nothing to report after execution (for example, if an object has been deleted).

Error type

Field	Type	Required	Description
type	string	No	Defines an error sub-type that allows for specialization of the error messages within the status codes
message	string	Yes	Human-readable description of the error that occurred