



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.

# Genesys Intelligent Automation Reference Guide

Using WebIVR MicroApps

---

## Contents

- 1 Using WebIVR MicroApps
  - 1.1 Prerequisites
  - 1.2 Configuring your environment
  - 1.3 Using MicroApps
  - 1.4 Detailed process summary

# Using WebIVR MicroApps

This page describes how you can use WebIVR-based MicroApps in chat interactions to accomplish various business tasks. For example, if an agent is helping a customer with a purchase and you want to securely collect the customer's credit-card information, you can use a MicroApp to securely capture the information without requiring it to go through the agent.

See the [Detailed process summary](#) section, below, for more information on the Intelligent Automation widget architecture and how the process executes among various Genesys components.

## Prerequisites

The following prerequisites are required:

- Intelligent Automation 3.6.x or higher with Messaging Server
- [Workspace Desktop Edition](#) or [Workspace Web Edition](#) (8.5.117.07 and higher)
- [Genesys Widgets](#)
- The following [eServices](#) components:
  - Chat Server (8.5.104.10 and higher)
  - Interaction Server (8.5.109.01 and higher)
  - Knowledge Manager (8.5.x and higher)
  - Genesys Mobile Services (8.5.107.15 and higher)
- [Configuration Server](#) (part of Management Framework) (8.5.101.08 and higher)

## Configuring your environment

### Update Genesys Widgets framework for Intelligent Automation

#### Important

The [Genesys Widgets](#) framework must be deployed in your environment before completing this section.

1. Go to the Intelligent Automation installation folder (for example, **C:\GAAP\Platform\TomcatMessaging\webapps\fish-messaging\widgets**).

2. Copy the latest widget JavaScript prefilters and CSS files from the Intelligent Automation folder noted in [Step 1](#) to your website.
  - a. Add **cx-speechstorm.css** to the folder containing other style sheets used within your website.
  - b. Add **cx-speechstorm.js** to the JavaScript folder containing **widgets.min.js**.
3. Integrate the Intelligent Automation widgets into your existing website. Refer to **index.html**, which is found in the Intelligent Automation folder noted in [Step 1](#), as a guide. This file contains various imports for the style sheets and JavaScript files that you copied in the previous step, along with the Standard CX Widget Instrumentation Script (which you should already have running as part of the Widgets framework).
4. In the file **index.html**, copy the script block that contains **sChatServerUrl** and **sSpeechStormServer** variables. You must update these variables to point to valid servers within your business. Verify all paths for imports are correct, according to the locations into which you copied files in the previous step.
  - a. Update **sChatServerUrl** to point to your chat server. For example:

```
var sChatServerURL = "http://<your_server>/gms_port_8010/genesys/2/chat/customer-support";
```

- b. Update **sSpeechStormServer** to point to your Intelligent Automation Messaging Server. For example:

```
var sSpeechStormServer = "http://<Messaging_Server:Port>";
```

## Add default server settings in Intelligent Automation

Next, you must log in to Intelligent Automation and add default server settings for use with the widget framework:

1. Log in to Intelligent Automation.
2. Go to **Administration > Default Server Settings**.
3. Configure the following server settings as necessary for your environment. In particular, ensure you set the correct host and port for Configuration Server, as defined in **GenesysSDK.ConfigServer.server.host** and **GenesysSDK.ConfigServer.server.port**.

GenesysSDK.ConfigServer.ClientApplicationName	default	remove
GenesysSDK.ConfigServer.LoginAsApplication	false	remove
GenesysSDK.ConfigServer.Password		remove
GenesysSDK.ConfigServer.Server.Host		remove
GenesysSDK.ConfigServer.Server.Port	2020	remove
GenesysSDK.ConfigServer.Username	demo	remove
GenesysSDK.InteractionServer.ClientName	GAAP	remove
GenesysSDK.InteractionServer.MediaType	chat	remove
GenesysSDK.ServerCommunication.AttachedData.IgnoreFailures	false	remove
GenesysSDK.ServerCommunication.ConnectionTimeoutMillis	9000	remove
GenesysSDK.Widgets.ProgressNotifier.Nickname	MicroApp	remove

4. Open Windows Services and restart Messaging Server.

## Using MicroApps

To use a MicroApp, an agent in a chat interaction with a customer enters a MicroApp URL into the chat. This chat message sets which MicroApp is used, and defines variables that Chat Server passes to the WebIVR application that powers the MicroApp. For example:

```
microapp://app/T3NB68E-/Payment  
AccountNumber=1234567  
Amount=USD77.80
```

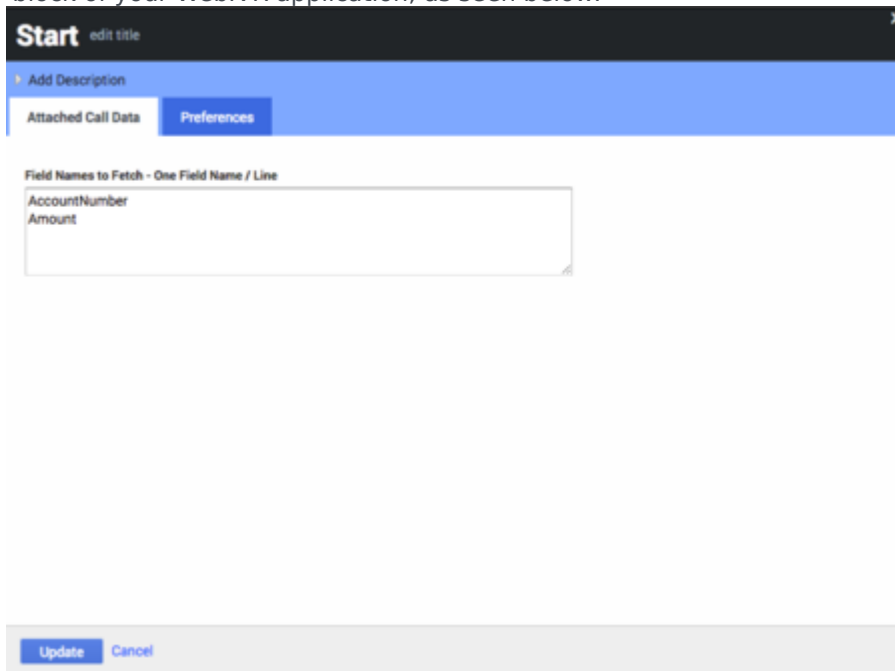
In the example above, the following parameters are set:

- **T3NB68E-** - This is the WebIVR URL token that is unique to this particular WebIVR application. To find the token for your WebIVR application, open your WebIVR application in the Callflow Editor and click the **Application Details** tab. This tab displays two tokens - one for the test version of your application and one for the production version.

The screenshot shows the 'Application Details' tab in the Callflow Editor. The top navigation bar includes 'App Automation Platform', 'Dashboard', 'Applications' (selected), 'Integration', 'Reports', and 'Personas'. Below this, a secondary bar contains 'Callflow Editor', 'Prompt List', 'Application Details' (selected), 'Deploy to Production', and 'Opening Hours'. A 'Currently Editing' dropdown menu shows 'Postcode'. The main content area is titled 'Application Details' and contains several sections: 'Application Name' with a text input field containing 'Postcode'; 'Application Description' with a large text area; 'Web IVR URL Token (Test)' with the value 'T3NB68E-'; 'Web IVR URL Token (Production)' with the value 'hzK+Rc0-'; 'Callflow Editor' with a dropdown menu set to 'Graphical'; 'Options' with checkboxes for 'Callflow Locked' and 'Public'; 'Personas Supported by this Application' with the text 'All personas supported (show details)' and a checkbox for 'Apply these Personas to all this Application's submodules'; and 'Module Parameters' with a checkbox for 'Make this Module Parameterisable'.

- **Payment** - This parameter is optional. It allows you to provide a name that describes the purpose of the widget to be launched.

- **AccountNumber=1234567** and **Amount=USD77.80** - These are parameters that Workspace passes into the Intelligent Automation widget. In this case, it specifies the **AccountNumber** variable has a value of **1234567** and the payment **Amount** is **USD77.80**. You define these variables in the **Start** block of your WebIVR application, as seen below:



### Important

Contact your Genesys representative for more information on parameters you can use with your widget implementation.

## Use Knowledge Manager to set up standard responses

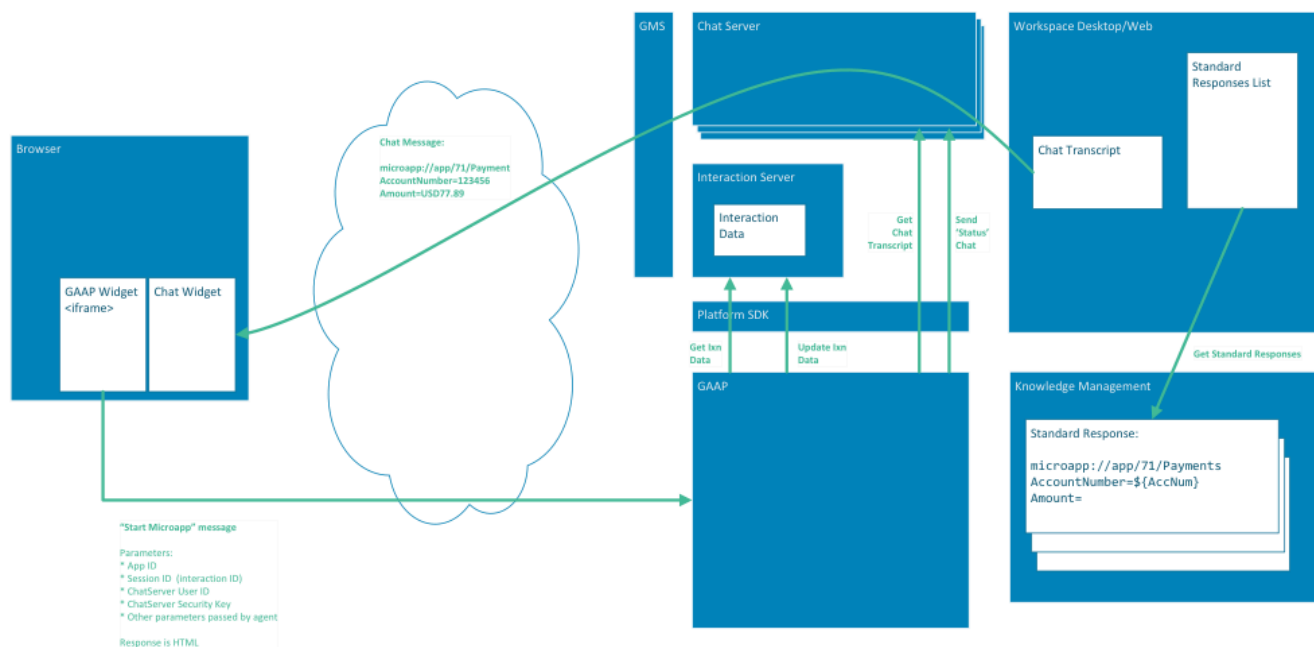
For a more efficient workflow, you can use Knowledge Manager (part of eServices) to set up standard responses that agents can insert into chats to invoke MicroApps.

Refer to the [Genesys Knowledge Manager](#) documentation for more information.

## Detailed process summary

This section describes in more detail how MicroApps interact with Intelligent Automation and the Genesys environment.

The following graphic explains the high-level architecture of how MicroApps interact with other Genesys components:



The list below provides a step-by-step account of how Intelligent Automation interacts with the Genesys environment to provide the MicroApp service.

1. The agent invokes the MicroApp URL and parameters in a chat window with a customer, preferably using a standard response defined in Knowledge Manager.
2. After the agent sends the URL to the customer, the Intelligent Automation widget uses a pre-filter to prevent MicroApp messages from displaying in the customer's chat window.
3. The Intelligent Automation widget intercepts the *microapp request* message and forwards the request in the correct format to the Intelligent Automation Messaging Server, along with the session ID (also known as Interaction ID), the Chat Server user ID and security key for that session, the MicroApp App ID, and any additional parameters provided by the agent.
4. When the MicroApp session begins, the Intelligent Automation Messaging Server contacts Interaction Server to retrieve any attached data already present in the interaction. This data is made available to the MicroApp in the same way as SIP Server attached data is made available in a voice call.
5. Intelligent Automation Messaging Server contacts Chat Server with the user ID and security key, and retrieves the chat transcription for this session.
6. After receiving the chat transcription, the Intelligent Automation Messaging Server finds the matching *microapp request* message in the transcription to verify that parameters have not been changed by the customer or a third party.
7. As the MicroApp progresses, the Intelligent Automation Messaging Server send *status* chat messages to the agent. These messages state the location of the customer in the MicroApp. The Intelligent Automation widget filters these status messages so that they do not appear in the customer's chat window.
8. The MicroApp might use logic to attach data to the interaction. If so, the Intelligent Automation Messaging Server sends messages directly to Interaction Server. This results in an *interaction data update* notification in Workspace and updates to values in any Case Information fields. For example, you can use this functionality to notify agents when the caller has identified himself or herself.
9. After the MicroApp ends, the Intelligent Automation widget sends a pre-filtered *status* chat message to

the agent on behalf of the user to signify the MicroApp portion of the interaction is complete.

### Important

During the whole interaction described above, the agent and customer can send normal chat messages to one another without affecting the MicroApp's execution.