

US4826783

Capability to move members from Voice channel to OmniChannel

Project Nickname: Voice2Chat

0.1  
9/13/2022

Contents

[1 Introduction 0](#_Toc114002608)

[2 Voice Member experience 1](#_Toc114002609)

[3 High level Technical Diagram 3](#_Toc114002610)

[4 Flows Diagram 4](#_Toc114002611)

[5 Functionality 6](#_Toc114002612)

[5.1 Rules 6](#_Toc114002613)

[5.2 PreChat 6](#_Toc114002614)

[5.3 Member 6](#_Toc114002615)

[5.4 SMS 6](#_Toc114002616)

[5.5 GMS – Create Storage 6](#_Toc114002617)

[5.6 Widget 7](#_Toc114002618)

[5.7 GMS – Get Storage 7](#_Toc114002619)

[5.8 Chat session 7](#_Toc114002620)

[5.9 Stargate 7](#_Toc114002621)

[6 Service description 8](#_Toc114002622)

[6.1 PreChat 8](#_Toc114002623)

[6.2 Create Storage 8](#_Toc114002624)

[6.3 Get Storage 9](#_Toc114002625)

[7 Genesys Rule Engine 10](#_Toc114002626)

[8 KVPs 11](#_Toc114002627)

[8.1 Voice KVPs: 11](#_Toc114002628)

[8.2 Portal widgets KVPs: 11](#_Toc114002629)

# Introduction

For a true OMNI channel solution, we should be giving the member the ability to move between channel types. Today we have the Callback solution that is used while queuing an inbound voice call that will move the member interaction from voice to callback channel. The Webcallback – is a feature, where a member can request a callback from a chat and that will also move the member interaction from Digital (Web Page or Chat) to Voice channel.

In this document, we will describe how we are moving further with a more OMNI channel feature, where a member waiting in a voice queue will be able to switch from Inbound Voice to a Chat channel.

This solution is a multichannel solution, several channels will be involved in a simple scenario:

* Voice – Member calls and is waiting in queue
* SMS – Sending the Chat link to the member
* Widget – User interface for a member
* Chat – Member switches from the Voice to Chat channel

In this document, we will describe how the Voice2Chat functionality.

# Voice Member experience

For this Voice2Chat to be triggered in the workflow, the Voice2Chat Feature should be enabled in GRAT.

If the feature is enabled, the Voice2Chat flow will start in the following two cases

## HOOP Closed

Members that are calling outside Voice hours of operation and within the Chat hours of operation and there are chat agents available to handle the members chat request, the member will have the ability to switch to chat.



## In-Queue

Members waiting in the Queue, in addition to callback offering, we will offer a chat with an agent.

Below is the high-level diagram on how the member’s experience will look like while waiting in the queue for an available agent



Triggering Voice2Chat functionality in the callflow should be similar to the way we enable callback offering.

If the Chat offering functionality is switched on, the Voice workflow will call the PreChat service to find out if the Chat HOOP is open and if Chat agents are available, then we will offer the member to start a chat instead of waiting in the voice queue.

If the member accepts to move further with a chat session, the voice IVR will collect the member’s consent to send an SMS that includes the Chat URL. Also, we will collect the member’s phone number to send the SMS to. After the SMS is sent, the voice application will hang up the voice call.

One sub-workflow should gather all the new functionality to make it easy to reuse this part of the workflow in other scenarios in the future.

# High level Technical Diagram



# Flows Diagram

Flow diagram shows processing of Voice2Chat processing requests.



# Functionality

Members should be able to switch from the telephony voice channel to chat channel.

Before offering such switch, the below elements should be taking into consideration:

1. Check Operational hours
2. Chat Agent availability

## Rules

To enable this functionality, the feature Voice2Chat should be enabled

There should be t new Features

1. Voice2Chat if Voice HOOP is closed (Boolean)

Here we should be able to enable or disable Voice2Chat

Default value: check box checked

A parameter within the same feature should also be defining how long should the GMS storage be available (integer)

Default value: 900

One more parameter will control if the widget forces the customer to authenticate before waiting in the Chat Queue. Some LOBs will not require Authentication, other LOBs have authentication as mandatory. (Boolean)

Default: Check box not checked

At the end. the feature should be something like:

**2.Feature Activation – Voice2Chat if HOOP Closed is active {pFlag} Storage is active for {integerValue} seconds. Widget Authentication enabled {pFlag2}**

1. Voice2Chat in Queue (Boolean)

Here we should be able to enable or disable the feature when waiting in voice queue.

In this feature, we need to submit an extra parameter “threshold” in seconds. (integer)

Default value: 900

If the feature is enabled AND the Voice estimated waiting time is bigger than the threshold, Voice2Chat will be triggered.

An example of how the feature will look like:

**2.Feature Activation – Voice2Chat In-Queue is active {pFlag} Threshold {IntegerValue2} Storage is active for {integerValue} seconds. Widget Authentication enabled {pFlag2}**

## PreChat

If the Voice2Chat feature is enabled, the **PreChat** request will be initiated from the Voice IVR.

PreChat is a web request that includes the Unit, Segment and Function as input parameters. The Chat Unit Segment and Function in the PreChat Solution does not have to be the same as the Voice Unit, Segment and Function.

PreChat service already exists today, no changes will be needed in this part.

## Member

Based on the PreChat response and availability, we will offer the member to switch to chat channel and the member will be able from the Voice IVR to accept or reject.

## SMS

If the member chooses to switch from Voice to Chat, a Chat URL will be needed. To send the URL to the member, we need to collect the member’s consent and to confirm the number to send the SMS to. The URL will include a storage key that will be used to fetch Interaction data from GMS (described below)

## GMS – Create Storage

To transfer the interaction information from the Voice environment to the Chat environment, we will be creating a record (storage) in the GMS for a specific amount time. Each record must include the following keys:

* ENT\_Unit
* ENT\_Segment
* ENT\_Function
* ENT\_ContactFirstNm
* ENT\_ContactLastNm
* ENT\_ReasonOfCall
* ENT\_ForceAuthentication

## Widget

When the customer visits the Chat URL that is sent by SMS, the GMS storage we created in the Voice environment will be fetched and used to start a chat interaction. All the storage response elements should be added to the ChatStart URL to attach these elements as UserData to the chat interaction.

In case the customer visits the Chat URL, but the GMS storage is expired and does not exist in the GMS anymore, the widget should present a widget where new chat interaction can be submitted. In this case, none of the data from the Voice interaction will not be used.

## GMS – Get Storage

Using the Storage Key parameter in the chat URL, a Get Storage service will be called to get all the data needed.

The data fetched from the GMS storage will be used as input parameters to create the chat session.

## Chat session

The widget will start a chat session using the data fetched from the GMS storage. No Changes will be needed in the chat routing project.

## Stargate

To access the GMS Create Storage and Get Storage services from a DMZ, we need to create entries in the stargate. This will be deployed only in case of Voice2Chat service should be used by Chatbot or third party chat clients.

# Service description

This section describes functionality of the Voice2Chat service.

## PreChat

* Request

[https://dev2-gms.uhc.com/genesys/1/service/DgtOPTOptumCareServicePrechat?\_ENT\_Segment=OMNI&\_ENT\_Function=DemoDigital&\_ENT\_Unit=Enterprise](https://dev2-gms.uhc.com/genesys/1/service/DgtOPTOptumCareServicePrechat?_ENT_Segment=OMNI&_ENT_Function=DemoDigital&_ENT_Unit=Enterprise&_email=k_suresh1@optum.com&_ENT_ContactFirstNm=Suresh&_ENT_ContactLastNm=Kumar)

* Method

GET

* Response

    "serviceStatus": "success",

    "serviceFailureReason": "",

    "hoursOfOperationStatus": "closed",

    "closedReason": "noagents",

    "holidayname": "",

    "hoursOfOperation": "Mon-Sun:12:00AM-11:59PM",

    "longestWaitTime": "-1",

    "longestWaitTimeThreshold": "300",

    "chatsInQueue": "-1",

    "chatsInQueueThreshold": "50",

    "agentsLoggedIn": 0,

    "agentsLoggedInThreshold": "0",

    "agentsReadyChat": 0,

    "cacheMaxAge": "180",

    "ChatType": "OPT\_OptumCare\_Service\_Chat",

    "VAG": "",

    "VQ": "",

    "debugMessages": "",

    "timezone": "US/Central"

## Create Storage

* URL

/genesys/1/storage/custom/{customId}/{ttl}

* Method

POST

* URI Parameters

1. customId (string)

Should be the Voice Orchestration Session ID. The same customId will be used to form the Chat URL to be sent to the member via an SMS.

1. ttl (number)

The time to live for this data, specified in seconds. The data is automatically deleted after it has been stored for {ttl} seconds. The ttl must be greater than zero (0). If an incorrect value is specified, a default of 30 seconds is defined.

* Example

<https://dev2-gms.uhc.com/genesys/1/storage/custom/33224050088888483375/900>

* Request Body (A MultiPart form or a URL encoded form consisting of different items representing the key/value pairs to store)
* ENT\_Unit
* ENT\_Segment
* ENT\_Funfction
* ENT\_ReasonOfCall
* ENT\_ContactFirstNm
* ENT\_ContactLastNm
* ENT\_PhoneNumber
* ENT\_ForceAuthentication

## Get Storage

* URL

/genesys/1/storage/{id}

* Method

GET

* URI Parameters

id (string) Should be the Voice Orchestration Session ID

* Example

<https://dev2-gms.uhc.com/genesys/1/storage/33224050088888483375>

* Response

    "ENT\_Function": "DemoDigital",

    "ENT\_ContactLastNm": "Mohamad",

    "ENT\_Unit": "Enterprise",

    "ENT\_Segment": "OMNI",

    "ENT\_ContactFirstNm": "Ali",

    "ENT\_PhoneNumber": "123456789",

    "ENT\_ReasonOfCall": "payment missing"

## Send SMS

* URL

<https://dev2-secureemail.uhc.com/DgtSMSService/DEV2/v1/sendSms4Ivr>

<https://stg-secureemail.uhc.com/DgtSMSService/STAGE/v1/sendSms4Ivr>

* Method

POST

* Body

"\_outbound\_message\_body":"Test SMS text",

"\_outbound\_message\_from":"15852826817"\*

"\_outbound\_message\_program\_id":"498309"

\*\_outbound\_message\_from is the parameter that includes the mobile number we will send the SMS to.

* Response

{

"\_outbound\_message\_status" : "success",

"\_outbound\_message\_id" : "6085565",

"\_outbound\_message\_from":"2195721",

"\_outbound\_message\_program\_id":"498309"

}

# KVPs

There are 2 types of KVP sets

## Voice KVPs:

* WidgetHost

This KVP is the host name for the widget hosting the Voice2Chat session on the widget side. It will be used to form the URL to be sent to the member. Different values will be used in Dev2, Stage and Production

**Values for the different environments can requested from Ramkumar Swaminthan**

* VoiceORSSID

This KVP is the Voice Orchestration Session ID. It will also be used as a key to create and fetch the Storage from GMS. VoiceORSSID is the second parameter to be used to form the URL to be sent to the member.

The chat URL should look like:

https://{WidgetURL}t?GUCID={VoiceORSSID}

* + Chat\_ENT\_Unit

Should be in the Voice LOB JSON parameter. If exists, then the value of this parameter will be used to save the ENT\_Unit in the GMS Storage. If Chat\_ENT\_Unit does not exist in the JSON parameter, the Voice ENT\_Unit will be used.

* + Chat\_ENT\_Segment

Should be in the Voice LOB JSON parameter. If exists, then the value of this parameter will be used to save the ENT\_Segment in the GMS Storage. If Chat\_ENT\_Segment does not exist in the JSON parameter, the Voice ENT\_Segment will be used.

* + Chat\_ENT\_Function

Should be in the Voice LOB JSON parameter. If exists, then the value of this parameter will be used to save the ENT\_Function in the GMS Storage. If Chat\_ENT\_Function does not exist in the JSON parameter, the Voice ENT\_Function will be used.

* + ENT\_ReasonOfCall

This KVP will be used to give the Chat agent an idea about, why the member tried to call the contact center. The value of this key can be taken from the IVR menu choices, voice entry points. This can be decided by the Voice application developers. This KVP will be one of the parameters to be saved in the GMS storage.

* + ENT\_ForceAuthentication

The feature “Voice2Chat - Enable Chat Authentication (Boolean)” will be used to set the value of ENT\_ForceAuthentication. If enable, the widget will request the member to enter a user name and password before placing the chat in the chat queue.

## Portal widgets KVPs:

* GUCID

This KVP is taken from the Chat URL that the member has visited. The value of the parameter will be used to get the GMS Storage data.

* All the parameters fetched from the GMS Storage data will be used as input parameters when starting the Chat session. Below is a list of the mandatory parameters needed to start a chat:

ENT\_Unit

ENT\_Segment

ENT\_Function

ENT\_ContactFirstNm

ENT\_ContactLastNm

Below are the optional KVPs

ENT\_ReasonOfCall

ENT\_ForceAuthentication

* + - ENT\_MediaClassNm must be added as an input parameter to the chat session. The value must be “chat” and will be used in the chat routing application.