5.3 Exercises

Bright Pattern Documentation

Generated: 6/25/2021 6:21 am

Table of Contents

Overview

Scenario-building exercises are provided in order to guide you if you're creating scenarios from scratch. Examples are presented for both chat and voice scenario types. Additionally, downloadable templates are available for each exercise and can be imported to your Contact Center Administrator application for training purposes. Explore our examples to learn more about scenarios.

Exercises

- All Scenario Exercises
- Downloadable Scenario Templates
- Create a Basic Scenario
- Pop Case or Contact Information
- Chat Scenario That Uses Bots
- How to Configure a Chat Scenario That Uses a Microsoft Azure Web App Bot
- Block Specific Incoming Phone Numbers
- Distribute Surveys to a Percentage of Random Customers
- Route Callers to the Last Agent and Provide a Voicemail Option
- Sending Automatic Email Replies to Customers Who Use the "Leave a Message" Chat Form
- Skill-Based Call Routing with an Auto Attendant Choice
- Voice Scenario Survey
- Use Conversational IVR in a Scenario
- Scenario Example

Scenario Builder Basics

If you've never designed a scenario before, you've come to the right place! In this article, we will present some very basic scenario building principles as well as quick tips you will find yourself coming back to time and again.

Refresher: What is a Scenario?

As we learned in section Scenario Builder Overview, a scenario is the logic of automated interaction processing. A real-life example of this: When you call a business, you are greeted with an automated menu (IVR), and are then asked to select a number matching your language preference. After selecting your preferred language, you are connected to an agent who speaks this language. What a customer takes for granted as a standard phone interaction is, in fact, a carefully constructed interaction flow, designed with blocks that execute specific functions.

Scenarios can be as basic or as complicated as you would like them to be. No matter how you design your scenario, however, for Bright Pattern Contact Center scenario design, the following are some guiding points that will help you on your journey.

With time and patience, you can create scenarios as complex as this

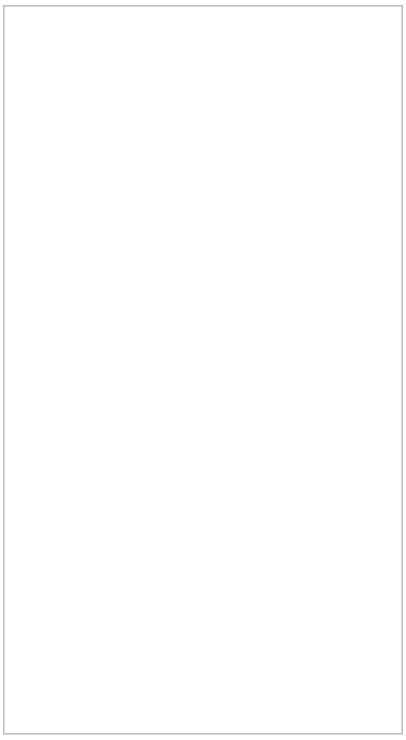
Important Scenario Blocks

Find Agent and Connect Call/Chat

Certain scenario blocks are meant to work in pairs. The first and most basic pair of blocks areFind Agent and Connect Call (if a voice scenario) or Find Agent and Connect Chat (if a chat scenario). These two blocks will be the basis for most of your scenarios.

So, what makes these blocks important? Find Agent does the heavy lifting of finding the appropriate person to connect the interaction to, and then Connect Call/Connect Chat makes the connection happen. It is because the Connect Call/Connect Chat block completes the action of connecting the customer to the found agent, we recommend arranging these two blocks together and not placing other blocks in between.

For more information, see section How to Create a Basic Scenario.



A basic scenario with Find Agent and Connect Call

Scenario Design Tips

Save Early and Often

As with any virtual creation, remember to save your work periodically! Systems can crash and your hard work may be lost, which is the last thing you want to happen.

Define Conditional Exits

As a reminder, conditional exits are the multiple paths that the scenario can take after processing a given block; they determine how your scenario responds to certain conditions during interactions.

Why do we want to define conditional exits? So that we know what should be happening in our scenarios at all times. For example, in a Connect Call block, if the conditional exit No Answer is reached (i.e., an agent does not respond to a call), but no further action is defined, your scenario could end here, leaving you with a disconnected and unhappy caller. If, however, you define No Answer with a Goto block, when No Answer is reached your scenario can loop back to an earlier point, keeping the caller on the line until someone answers.

On this note, you may not need something to happen at every conditional exit. In this case, we recommend you define the exit with the Exit block.

Watch Out for Infinite Loops!

Infinite loops are segments of a scenario that, once entered, continue to direct back to the same starting and ending points, creating a loop that doesn't end and/or prevents a clean exit.

What are some infinite loop causes? Infinite loops can happen with theStart Another Scenario block. That is, if a new scenario is triggered and this new scenario redirects back to the original in the wrong place, you may have an infinite loop on your hands. Additionally, not defining conditional exits can lead to infinite loops (i.e., omitting an Exit block in the wrong place).

When Scenarios Go Wrong: How to Troubleshoot Your Scenario

After you've created your scenario, you will want to test it out to see if it functions correctly. If it does, well done! If not, don't worry. Bright Pattern Contact Center contains built-in methods that can help you identify and debug the source of your problem.

Use the Internal Message Block

The Internal Message block sends an internal message to a system user, but for troubleshooting purposes, this block allows you to put checkpoints in your scenario. For example, if you define the various conditional exits of a block with Internal Message, you will receive a message every time a specific exit is taken. Internal Message is a good thing to include in your scenario if you find yourself wondering, "Did my scenario go through this branch?" or "What was my variable value in this branch?"

Scenario Steps in Interaction Records

Another great tool for you to use is Scenario Steps inInteraction Records. This section will take you through a scenario's path from start to finish and will show how it exited the various scenario blocks. Note that it takes 15 minutes from the point in time when the scenario executes to the point when it becomes available in Interaction Records (i.e., this information is not as immediately available as an Internal Message; you have to wait).

How Scenario Steps look in Interaction Records

All Scenario Exercises

Bright Pattern provides scenario exercises, which show you how to build a scenario for a specific service and purpose. These exercises provide detailed descriptions of which scenario blocks to use, how to configure their properties, and why you need them. Sometimes you need a practice run before you can build your dream scenario.

General

- Scenario Builder Basics
- Downloadable Scenario Templates
- Create a Basic Scenario

Chat

- How to Create a Chat Scenario That Pops Case or Contact Information
- How to Create a Chat Scenario That Uses Bots
- How to Configure a Chat Scenario That Uses a Microsoft Azure Web App Bot
- Pop Case or Contact Information
- Sending Automatic Email Replies to Customers Who Use the "Leave a Message" Chat Form

Voice

- How to Blacklist Specific Phone Numbers
- How to Configure Last-Agent Routing Using the Internal Database
- How to Create a Voice Scenario That Blocks Specific Incoming Phone Numbers
- How to Create a Voice Scenario That Distributes Surveys to a Percentage of Random Customers
- How to Route Callers to the Last Agent and Provide a Voicemail Option
- General Inbound Voice Scenario
- Redirect Calls Economically with a Single-Step External Transfer Option
- Skill-Based Call Routing with an Auto Attendant Choice
- Use Conversational IVR in a Scenario
- Voice Scenario Survey

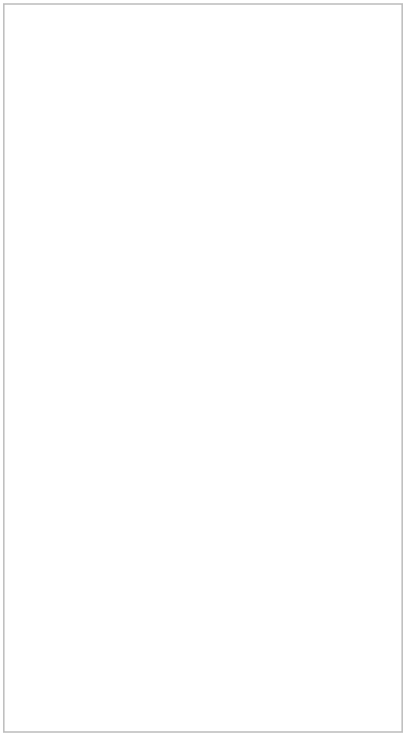
How to Create a Basic Scenario

This article describes how to create the most basic type of scenario, where the system finds an available agent and connects to a customer.

About

A basic scenario is a foundation upon which a more elaborate scenario can be built. Whether you are creating a chat or voice scenario, there are only two actions that need to occur: finding an agent and then connecting the agent to the customer. For a voice scenario, the blocks are Find Agent and Connect Call; for a chat scenario, the blocks are Find Agent (Chat) and Connect Chat.

Note that when you configure a basic scenario, the Find Agent block and Connect Call/Chat block should always be arranged back-to-back. The reason for this is the Find Agent block will find the agent but it does not open the communication channel; it sets the destination of the interaction using the $(destination) variable and then passes it to the Connect Chat/Call block, which opens the channel.



A basic voice scenario with defined conditional exits

Chat

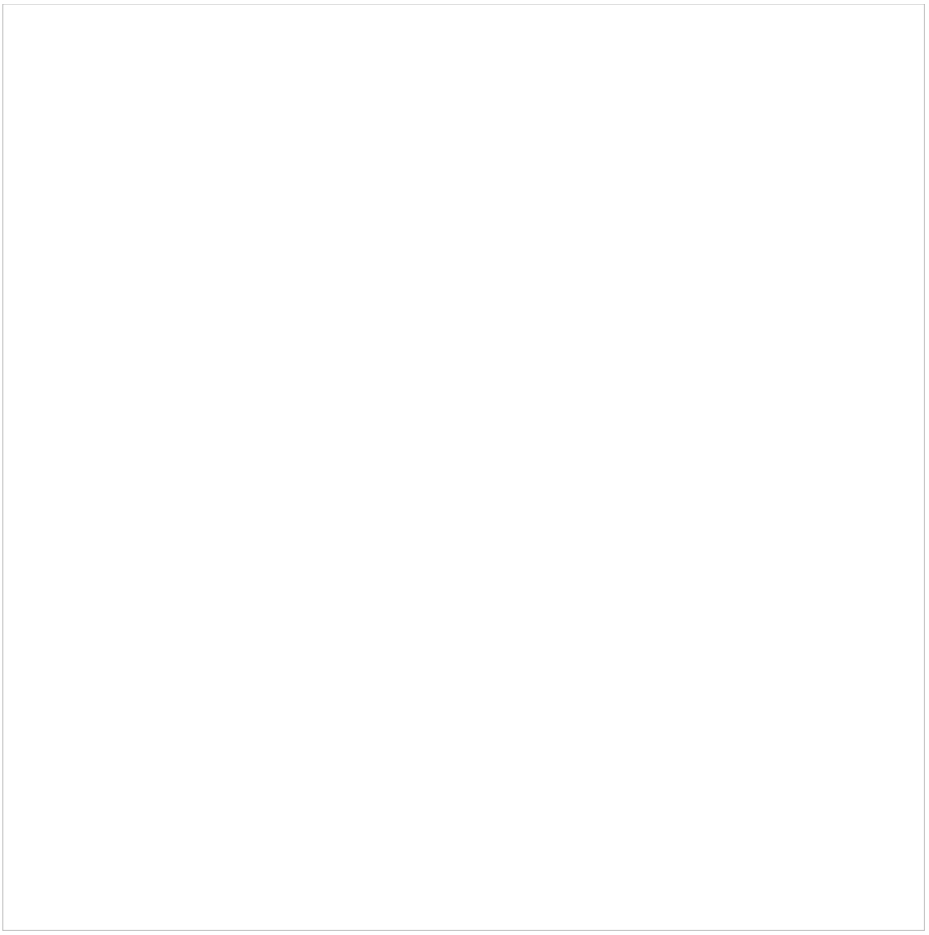
A basic chat scenario consists of the Find Agent (Chat) block and Connect Chat block.

Find Agent (Chat)

In the Find Agent (Chat) block, you will specify agent skills so the system knows which agents to find; also, you will specify the wait time and wait condition (i.e, omni-channel routing).

Additionally, you may add an interval. For example, you may want to specify that for the first 30 seconds, transfer the call to any agent with a service capacity of 25% or higher. Note that you can set multiple intervals to look for agents.

Also, it is possible to specify skills from different skill groups, and even multiple services, in only one Find Agent block (i.e., you do not need multiple Find Agent blocks).



The Find Agent block settings in a chat scenario

Connect Chat

The Connect Chat block allows you to specify a default service destination, an override service destination, and additional settings.

The Connect Chat block settings

Voice

A basic voice scenario consists of a Find Agent block and a Connect Call block.

Find Agent

Like Find Agent (Chat), Find Agent allows you to specify agent skills, wait time and condition, and transfer interval; however, there are additional settings for the voice elements associated with this block. Specific voice options include Virtual Queue and prompts.

The Find Agent block settings in a voice scenario

Connect Call

In the Connect Call block, you can set your destinations and other options such as custom hold music, service announcements, and so forth.

The Connect Call block settings

Suggested Reading

In order to better understand the basic principles of scenario design, we recommend reading the following articles:

- Scenario Builder Overview
- Scenario Builder Basics

And finally, don't forget to save your work early and often!

How to Create a Chat Scenario That Pops Case or Contact Information

The Agent Desktop application contains robust case and contact management features, allowing your agents to save, search for, and preview just about any information related to interactions; this information lives in internal databases.

Through scenarios, it is possible to make your extensive case and contact information instantly available to agents when interactions begin, all without your agents having to search for it. Not only will this help your agents save precious time, it keeps your returning customers happy, too.

This scenario example shows how, with only a few blocks, it is possible to pop an existing contact's case information to an agent.

How a saved contact's cases look when popped to an agent

Scenario Example

Click the following link to download an annotated version of this chat scenario example.
File:App Screen Pop Customer Cases.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

For general information about scenarios, refer to section Scenario Builder Overview.

Scenario Flow

The following chat scenario shows how a saved contact's email address is used to pop related case information to an agent; this is accomplished by using the Bright Pattern Search Object block, the If block, and the Web Screen Pop block.

Designer's note: Most of the configuration steps in this example can be applied in a voice scenario, albeit with the appropriate voice-related blocks and variables; in both chat and voice scenarios, the URLs required in the Web Screen Pop block are the same.

Finally, note that this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.

Scenario Overview

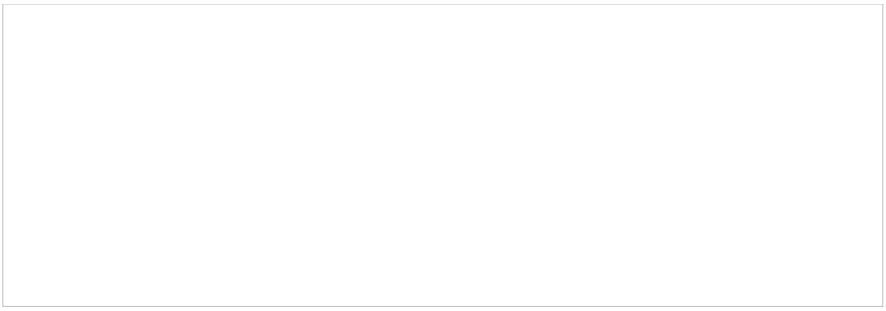
The diagram shown illustrates what the complete scenario looks like when designed in the Scenario Builder application.

Scenario-Pop-Case-Overview-53.PNG

Action 1: (Optional) Use Set Variable to Rename the Email Field

First, we use the Set Variable block to rename the variable *$(item.externalChatData.email)* as *chat.email*. Note that renaming $(item.externalChatData.email) is an optional step done to make the variable name easier to type in subsequent blocks.

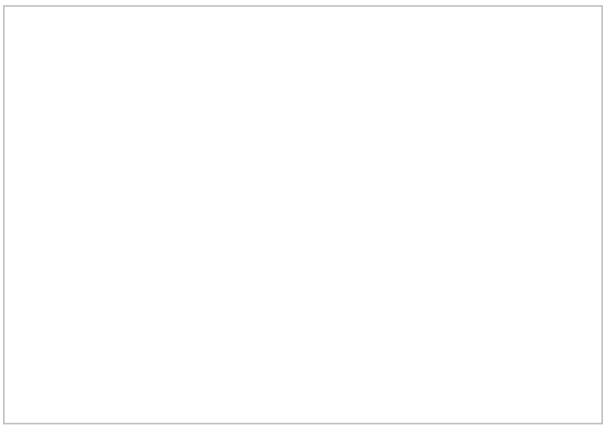
As a reminder, the variable $(item.externalChatData) pulls information from your configured chat widget; $(item.externalChatData.email) specifically references the email field we have configured in it.



Optional Set Variable block

Action 2: Search for a Matching Email

Next, we use the Bright Pattern Search Object block to search our internal database for an existing contact where the email address matches the value of the variable $(chat.email) (i.e., the email address entered in the chat widget). The results of the search are passed to the Recordset name field, which we name *email.RS*.



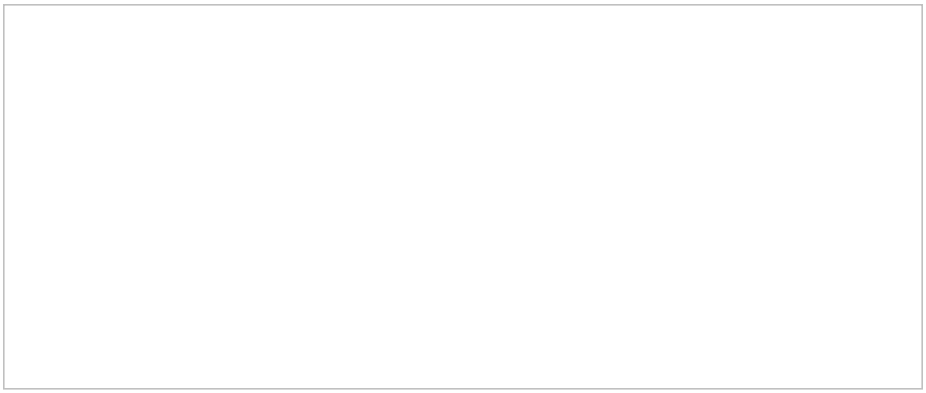
The Bright Pattern Search Object block searches for a matching email address

Action 3: Pass the Searched Email Address Results to an If Block

Next, we add an If block and configure it with two branches: *Email Found* and *Email Not Found*; the purpose of this is to be able to dictate different actions depending on whether the email address was found or not.

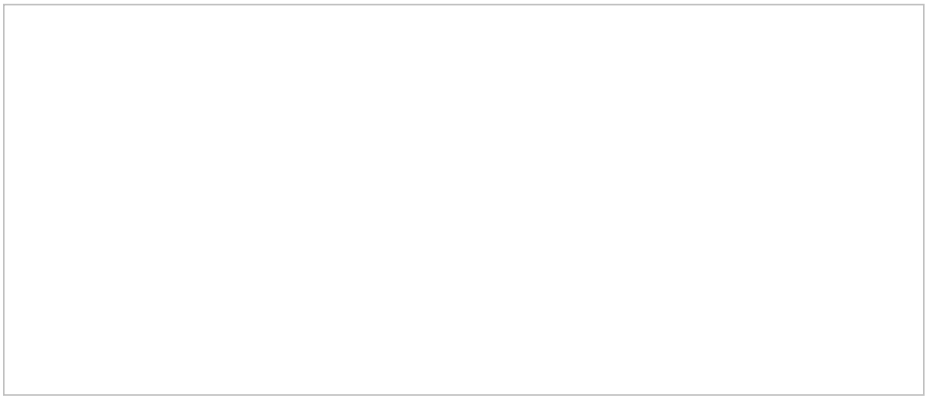
The If block with two branches

For the Email Found branch, add a condition to check if the string *email.RS* (i.e., the recordset name from Bright Pattern Search Object contains a value (i.e, "is not empty").



The If block's Email Found branch

For the "Email Not Found" branch, add a condition to check if the string *email.RS* does not contain a value (i.e., "is empty").



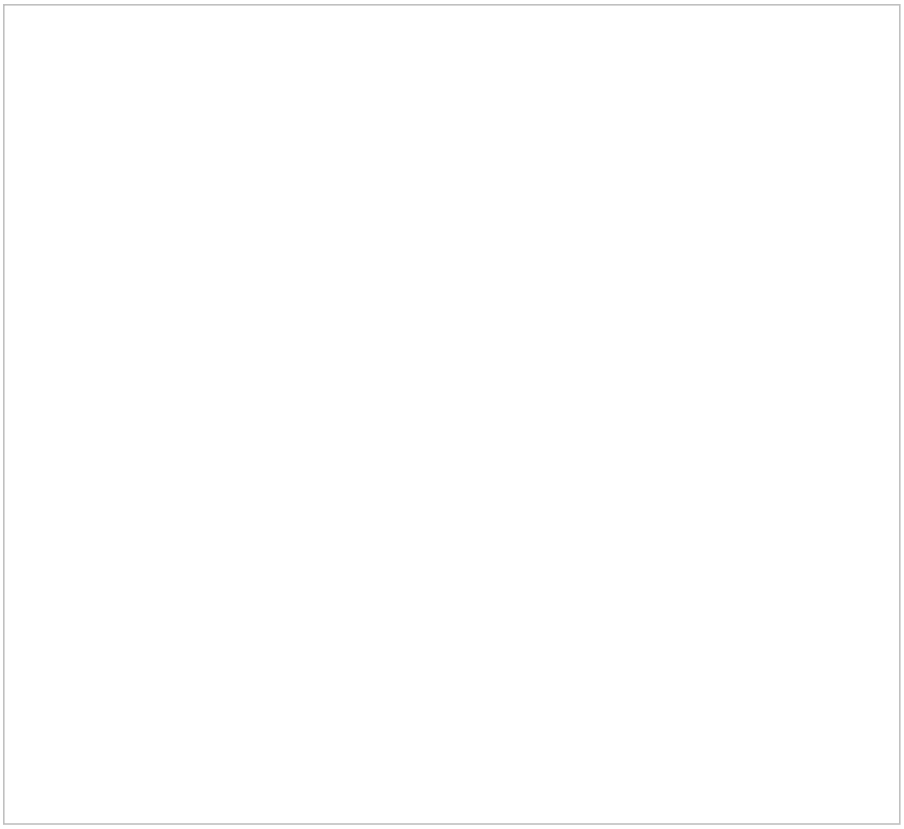
The If block's Email Not Found branch

Action 4: Configure a Web Screen Pop Block on the If Block's "Email Found" Branch

If a value was returned in the recordset name *email.RS* (i.e., the email address entered in the chat widget matches that of a saved contact), we want the cases associated with this contact to be popped to the agent from a Web Screen Pop block.

In this block, enter the following URL in the URL of the page to open field: https://your_contact_center.brightpattern.com/agentdesktop/contact/currentcontact/cases

From here, if the email address matches one associated with an existing contact, the cases associated with this contact will be popped to the available agent when the chat is connected. If not, no screen pop will occur. For more information on the types of URLs that can be popped, see section Access URLs in this article.



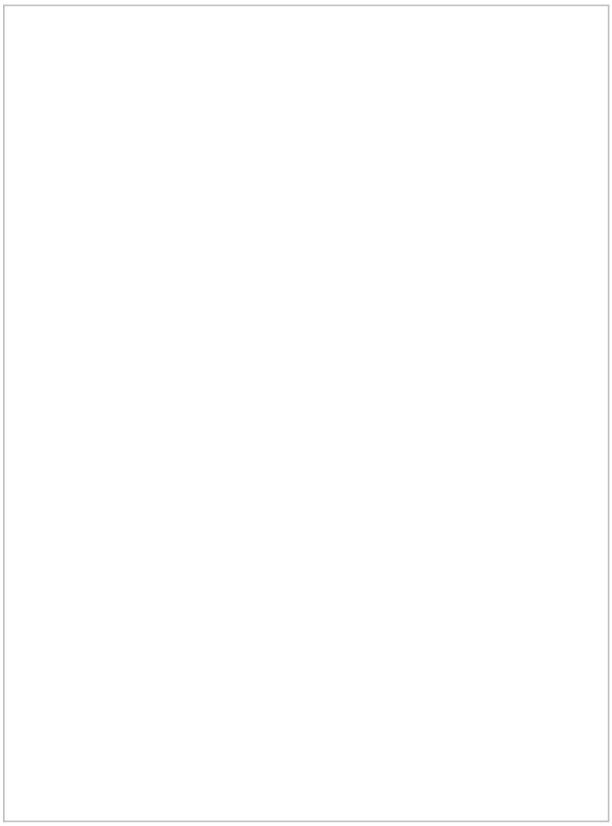
The Web Screen Pop block pops the cases associated with the found contact

Action 4a: Configure a Goto Block on the If Block's "Email Not Found" Branch

If no value was returned in *email.RS* (i.e., the email address entered in the chat widget does not match that of a saved contact), we want the customer to connect directly to an agent. To do this, we use a Goto block to jump ahead in the scenario to the Find Agent block.

Action 5: Finish with Find Agent/Connect Chat

Finally, we are ready to pass the customer to an agent, whether they were identified or not. To do this, configure your scenario with the two most basic chat scenario blocks: Find Agent and Connect Chat. For more information about these blocks, see How to Create a Basic Scenario. As a reminder, define all conditional exits.



Find Agent and Connect Chat with defined conditional exits

Access URLs

The following is a selection of URLs that can be used in theWeb Screen Pop block to pop specific case and contact information.

Case

The following URL can be used to pop case information.

Case Number

<your_contact_center>/agentdesktop/case/number/<case number>

URL Explained

- **<your_contact_center>** is your contact center's URL (e.g., https://some_name.brightpattern.com)
- **<case number>** is the specific case number

Contact

The following URL can be used to pop contact information.

Contact Tabs

<your_contact_center>/agentdesktop/contact/currentcontact/<details|activities|pending|cases>

URL Explained

- **<your_contact_center>** is your contact center's URL (e.g., https://some_name.brightpattern.com)
- **<details|activities|pending|cases>** are the options available to add to the end of the URL (e.g., https://some_name.brightpattern.com/agentdesktop/contact/currentcontact/details); these are the tabs found in every contact record.

How to Create a Chat Scenario That Uses Bots

To use a bot such as Watson Assistant in Bright Pattern chat interactions, you simply need to modify a chat scenario to include bot-specific scenario blocks such as Chat Bot Select Account and Ask a Bot.

This article will show you how to create a general chat scenario that uses a bot to:

- Automatically accept the customer's chat request
- Greet the customer
- Provide up to two answers to the customer
- Connect to an agent when it's determined that the bot is not being helpful
- Provide suggested replies to the agent during the chat

Prerequisites

If you have not already done so, please complete these steps before proceeding:

- Create a Watson Assistant
- Add a bot/chat suggestions engine integration account
- Download and import our bot scenario template: File:App Example Bot Scenario.zip

Scenario Flow

Scenarios are made up of *scenario blocks* that specify how interactions are processed. Detailed descriptions of Bright Pattern scenario blocks are available in this guide's *Scenario Block Definitions* section.

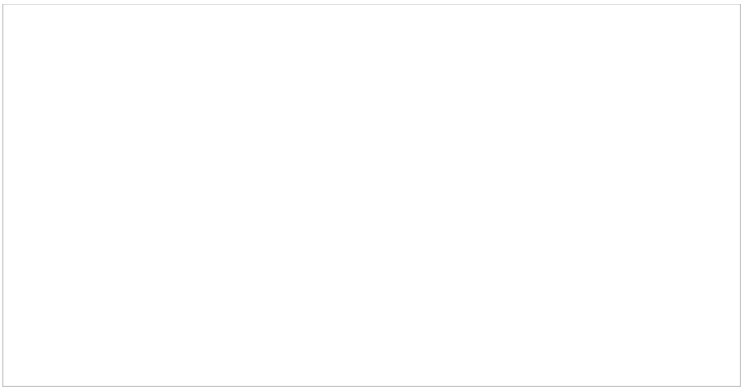
The following sequence of actions illustrates how scenario blocks function in a bot-driven chat scenario. This example scenario uses an integrated Watson Assistant.

Example chat scenario in Scenario Builder

Action 1: "Chat Bot Select Account" specifies the integration account for your bot

All chat scenarios that incorporate bots will include the Chat Bot Select Account block. In this block, you are selecting a bot/chat suggestions engine integration account.

It's possible for your contact center to have many integration accounts, so it's important to specify which one will be used for this scenario. If you do not see the desired account listed, go back and add an integration account.



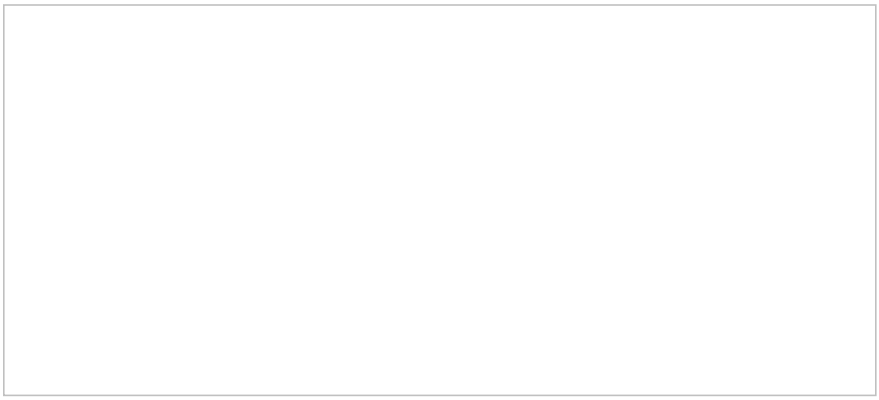
Select an integration account

Action 2: "Set Variable" personalizes the chat with customer data

In this example, the Set Variable block is used to get the customer's first name and last name.

Variable *$(item.externalChatData)* gets the customer's data from the pre-chat form that the customer completed before launching the chat. It's called external chat data because it comes from outside of the chat interaction.

To specify what type of external chat data you want, append *.first_name* or *.last_name* to the variable, like this:



Use a variable to get the customer's name from the pre-chat form and use it in the chat interaction

Action 3: Send Message puts the customer's name into a message

Now that you have a method to collect the customer's name, you can use theSend Message block to incorporate the name into a personalized chat message.

For example, "Hi *first_name*, how can I help you today?" becomes "Hi Jane, how can I help you today?"

Have the bot send a message to the customer, addressing him/her by name

Action 4: "Set Variable" defines scenario steps and tells the bot what to do at each step

We want the bot to respond to the customer two times before connecting to the agent. Using Set Variable blocks, we can define each bot's reply as a step, and tell the scenario what happens after each step.

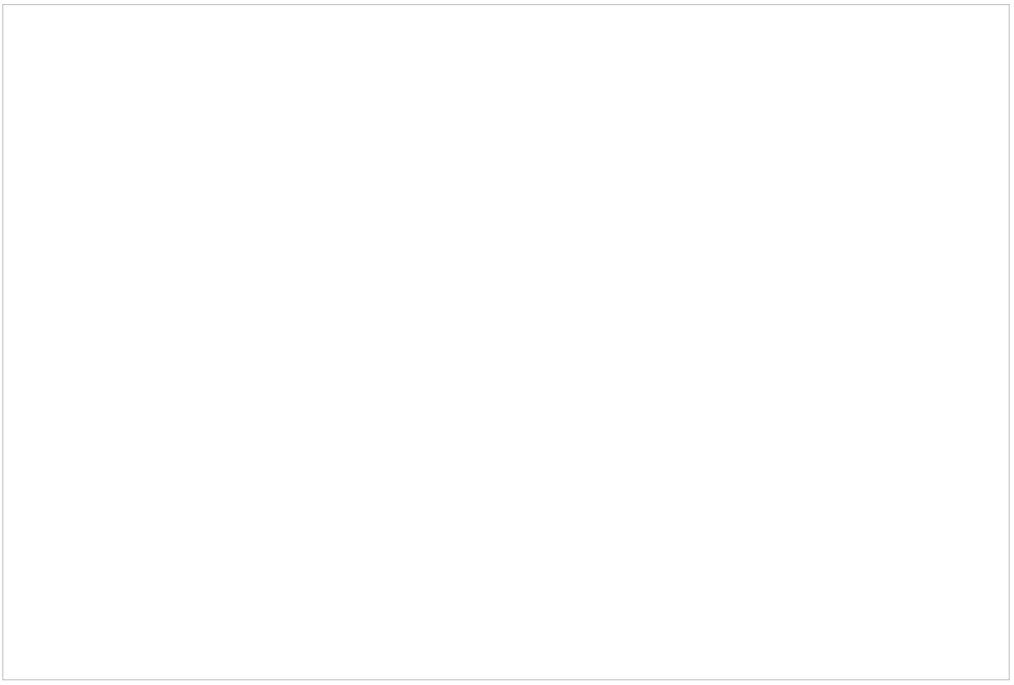
The first bot reply is specified as "step 0" and the second is specified as "step 1." The scenario will run through both steps. If the customer's issue is not resolved by step 1, the customer is transferred to the agent for help.

Use a variable to define this part as the first in a series of steps

Action 5: "If" tells the bot to send a message if the customer did not write a pre-chat form message

Most web chats are set up to begin with a pre-chat form that collects information from the customer (e.g., name, email address, phone number, reason for chatting, etc.) The If block specifies what happens if the customer did not write a message in the pre-chat form's Message field. If that field is empty, the scenario tells the bot to send a message.

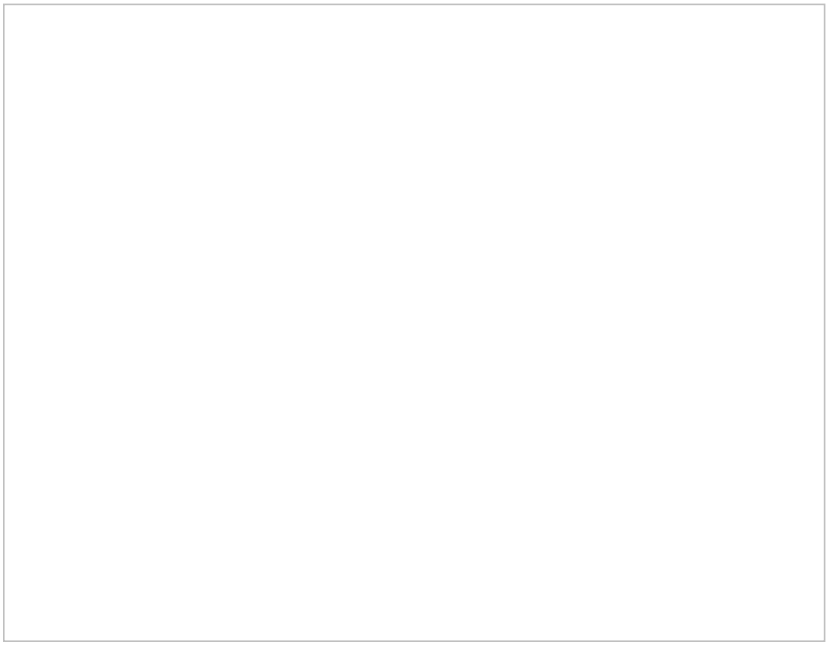
The message aligns with what you have configured your Watson Assistant to say. We call the bot's messages suggestions.



If there's no customer message, send a message

Action 6: "Ask a Bot" lets the bot talk to the customer

The Ask a Bot block allows the bot to automate the conversation before an agent is connected. It delivers the bot's raw response (i.e., suggestions) to the customer. A raw response is configured in your Watson Assistant dialog flow settings; it's unrelated to Bright Pattern.



Get suggestions from the bot

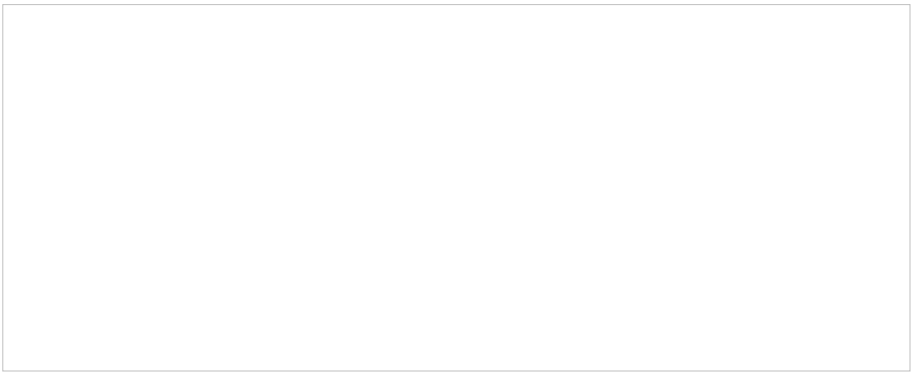
Action 7: "Send Message" sends a custom or system-wide message

Another message can be optionally sent to the customer at this time (e.g., "Do you want to talk to a real live person now?").

Send another message if you want

Action 8: "Set Variable" sets the next scenario step

This time, we are defining "step 1." Remember, bots are useful for self-service and routing, but they become annoying if they do not help your customer or meet expectations. Setting these scenario steps ensures that a customer is connected to an agent if the bot has not answered the customer's question by now.



Define this part as step 1

Action 9: "If" sets conditions that determine when the chat connects to agent

In this If block, a condition is set to make sure that if the scenario takes more than two steps to resolve the customer's issue, then it will transfer the chat from the bot to the agent.

If the bot is unhelpful, connect to agent

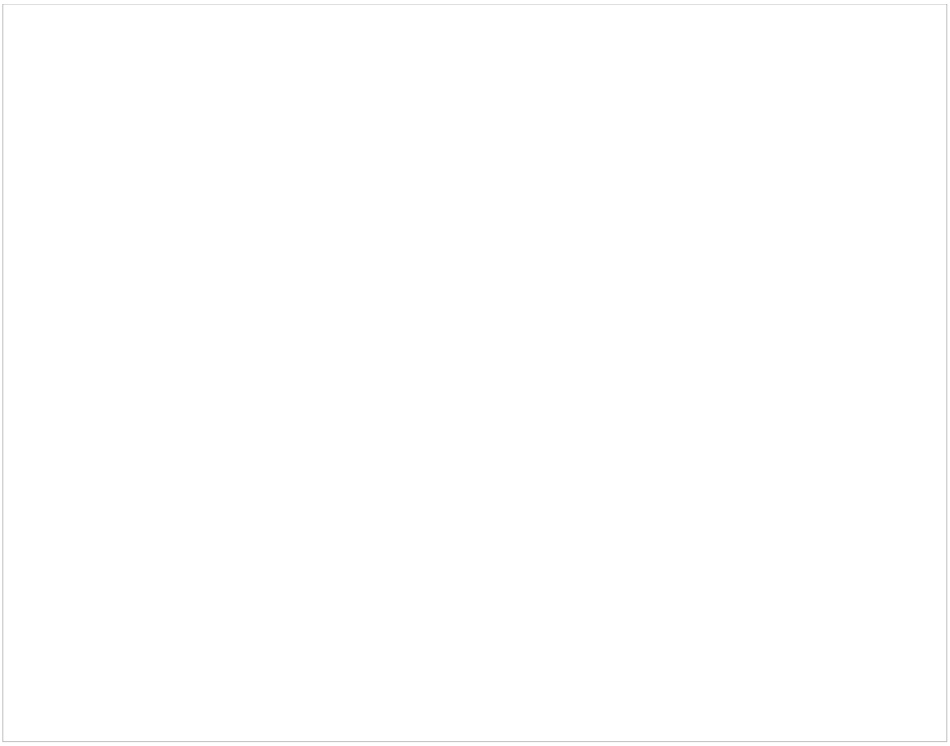
Action 10: "Find Agent" looks for a skilled agent

The Find Agent block looks for the next available skilled agent to accept the chat. You can use Find Agent to set wait times and send the customer messages about estimated waiting time (EWT).

The bot couldn't resolve the customer in two steps, so the scenario looks for an agent

Action 11: "Connect Chat" always comes after "Find Agent"

Although Find Agent uses criteria to identify the proper agent to receive an interaction, it does not have a "delivery" mechanism. Connect Chat is the delivery mechanism for the interaction.



You can either specify a destination for the chat or let the system find the next available agent

Action 12: "Exit" ends the scenario

The Exit block completes the scenario. Without it, the scenario will loop through this configured flow until the customer ends the chat.

Last Action: Save!

Be sure to click **Save** to apply your changes. Note that closing your web browser window or tab will close Scenario Builder without saving, and you will lose your work.

Then go try it!

Recommended Reading

For more information on chat configuration and bot setup, see these Bright Pattern tutorials:

- How to Configure Web Chat
- How to Create a Watson Assistant
- Scenario Builder Basics
- Tutorials for Admins > Chat Tutorials

How to Configure a Chat Scenario That Uses a Microsoft Azure Web App Bot

In our tutorial, How to Configure a Microsoft Azure Web App Bot for Chats, you learned how to set up a basic Web App Bot from a template.

In this scenario exercise, you will learn how to edit a Bright Pattern Contact Center scenario to use your Web App Bot in a chat conversation.

Procedure

Because we are using Bright Pattern scenario blocks for authentication to the Direct Line Bot Framework and for having a chat conversation, no integration account setup is needed. Simply download our example scenario, and follow the instructions to edit it.

Step 1: Import and edit example scenario

1. In the Bright Pattern Contact Center Administrator application, open the scenario that will be using your bot.

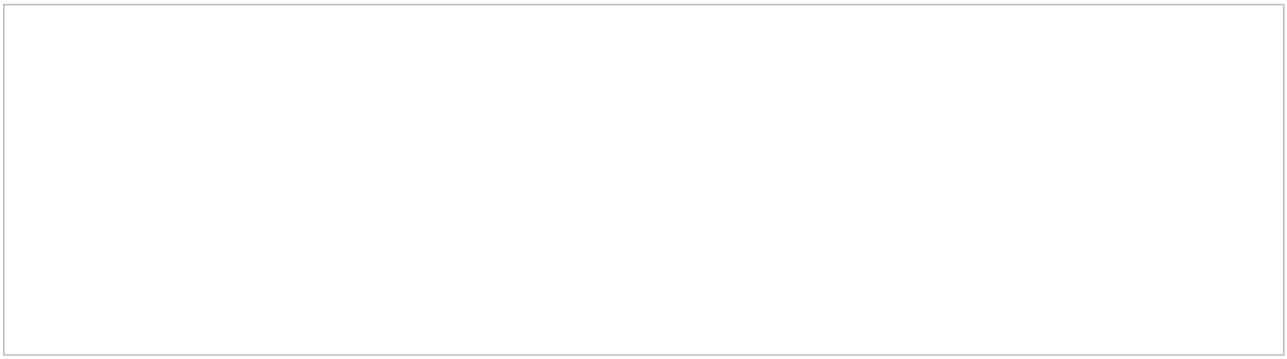
2. Import our example scenario File:App Azure Web App Bot Example.zip to get started quickly. This basic scenario includes comments to describe how the scenario blocks are being used.

Step 2: Add secret key in Set Variable

In the first Set Variable block, set the following properties:

1. **Variable name** - Name the block (e.g., “Secret”)

2. **Value** - The secret key generated by the Azure Bot Framework when you created the Direct Line channel. Your Bright Pattern scenario will use this secret key to authenticate the Direct Line API requests that it issues to communicate with your bot.



Variable scenario block with the secret key in the Value field

Step 3: Authenticate to Direct Line API 3.0

In the first Fetch URL block we will obtain a session token to authenticate requests to Direct Line API 3.0 by using the secret that you copied from the Direct Line channel configuration page.

Set the following properties:

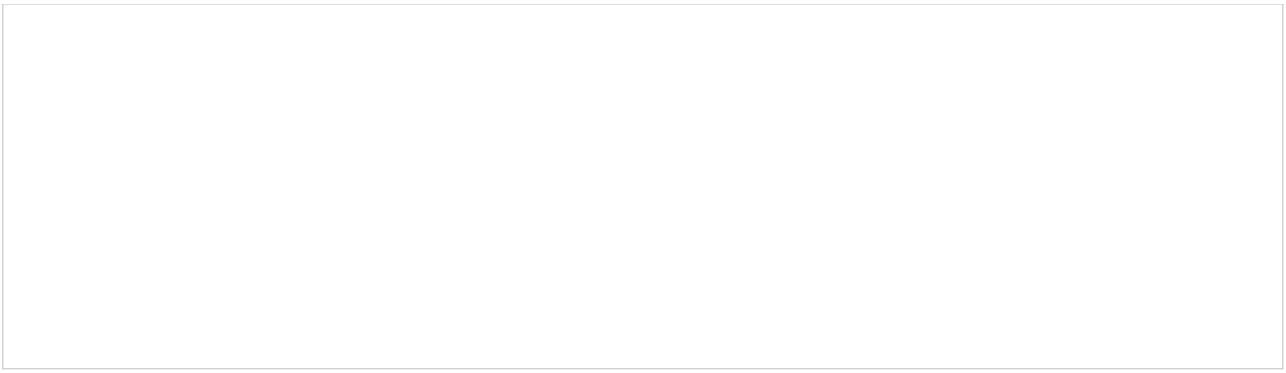
- **Title text** - Name the block (e.g., “Get token”)
- **Request type** - POST
- **URL to fetch** - https://directline.botframework.com/v3/directline/tokens/generate
- **Extra Headers** - Authorization: Bearer $(secret)
- **URL parameters** - None
- **Content Type** - application/JSON
- **Body** - Empty
- **Username** - Leave blank
- **Password** - Leave blank
- **Initial path in the result JSON** - Leave blank
- **Scenario variable prefix for JSON data** - jstoken

Fetch URL scenario block being used for authentication

Step 4: Add token to Set Variable

In the second Set Variable block, set the following properties:

- **Variable name** - Name the block (e.g., “token”)
- **Value** - $(jstoken.token)



Variable scenario block setting the token variable the Value field

Step 5: Start a conversation

In the second Send Message+ block, we will open a new Direct Line conversation by issuing a POST to the */v3/directline/conversations* endpoint. While the conversation is open, both the bot and client may send messages.

If the request is successful, the response will contain an ID for the conversation (i.e., "conversationId" which is needed in later steps), a token, a value that indicates the number of seconds until the token expires, and a stream URL that the client may use to receive activities via WebSocket stream.

Set the following properties:

- **Title text** - Name the block (e.g., "Start conversation")
- **Request type** - POST
- **URL to fetch** - https://directline.botframework.com/v3/directline/conversations/
- **Extra Headers** - Authorization: Bearer $(token)
- **URL parameters** - None
- **Content Type** - application/JSON
- **Body** - Empty
- **Username** - Leave blank
- **Password** - Leave blank
- **Initial path in the result JSON** - Leave blank
- **Scenario variable prefix for JSON data** - conversation

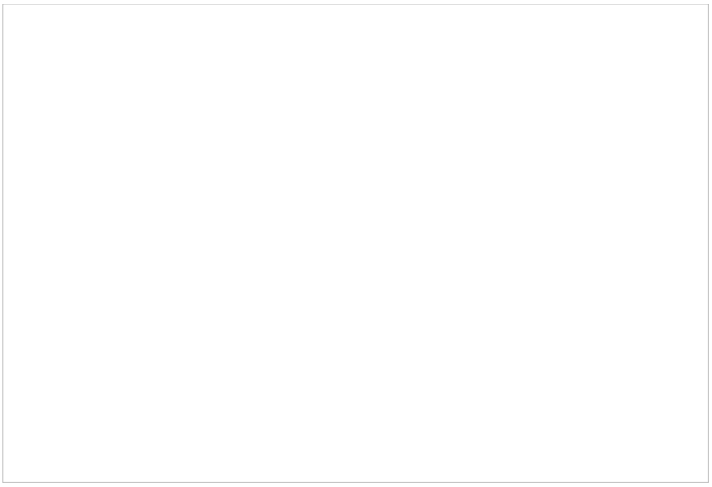
Fetch URL block being used to open a new Direct Line conversation

Step 6: Send a message to the customer

In the Send Message+ block, specify the message you want to send in the conversation.

Set the following properties:

- **Media type** - Chat
- **Message** - Any text message (e.g., “Hello and welcome!”)



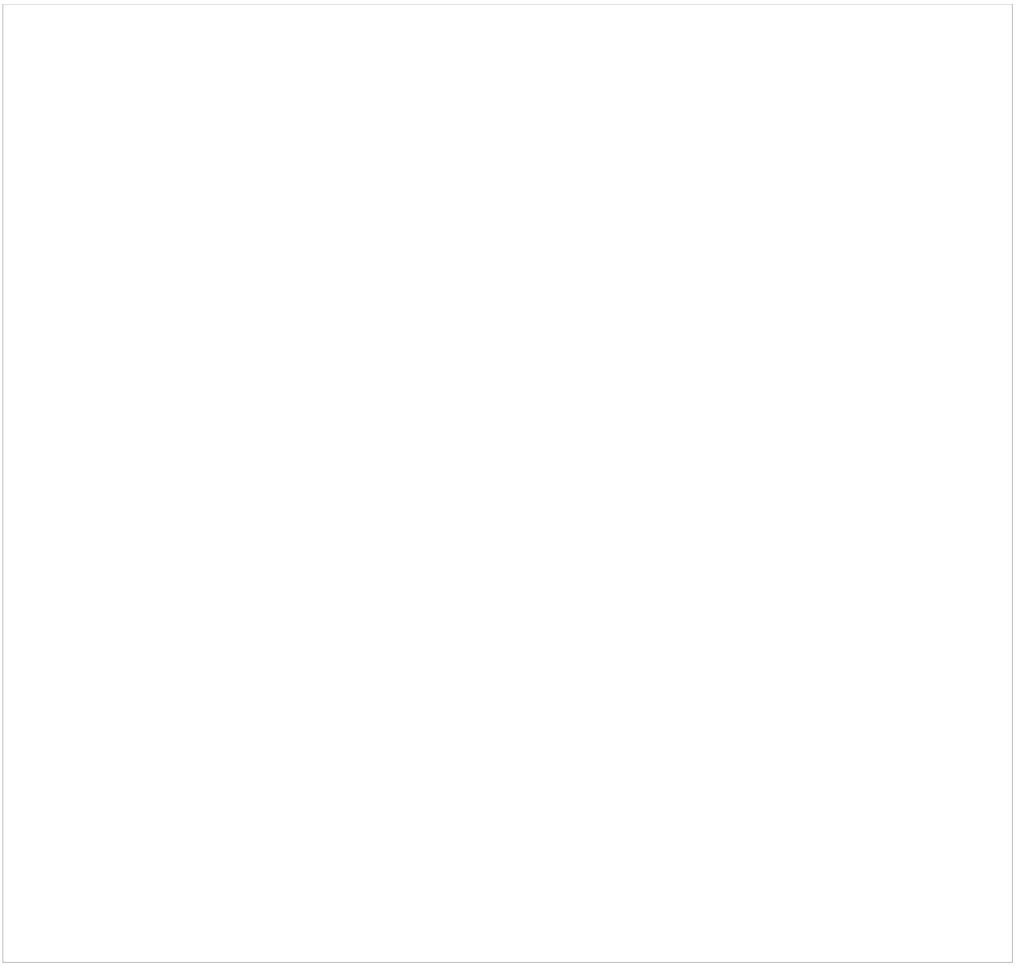
Send Message+ block being used to send a message to the customer

Step 7: Prompt for a customer response or wait

In the Request Input block, we can optionally prompt for a response from the customer and save the response to the "user_input" variable name. Alternatively, we can simply wait for text input from the previous scenario block.

Set the following properties:

- **Prompt with** - text message
- **Message to send** - Leave empty
- **Result variable name** - user_input



Request Input block could be used to prompt for a customer response

Step 8: POST activities

In the third Fetch URL block, we will use the Direct Line 3.0 protocol to exchange activities supported by your bot, like message activities and typing activities. We will issue a POST to the */v3/directline/conversations/$(conversation.conversationId)/activities* endpoint, where "$(conversation.conversationId)" is the ID that was obtained from the previous Fetch URL block's POST request to the the */v3/directline/conversations* endpoint.

Set the following properties:

- **Title text** - Name the block (e.g., "Post Activities")
- **Request type** - POST
- **URL to fetch** - https://directline.botframework.com/v3/directline/conversations/$(conversation.conversationId)/activities
- **Extra Headers** - Authorization: Bearer $(token)
- **URL parameters** - None
- **Content Type** - application/JSON

- **Body** - Specify the Activity object in the body of the request.

  For example:

```
{
   "type": "message",
   "from": {
      "id": "user1"
   },
   "text": "$(user_input)"
}
```

- **Username** - Leave blank
- **Password** - Leave blank
- **Initial path in the result JSON** - Leave blank
- **Scenario variable prefix for JSON data** - message_id

Fetch URL block being used to POST activities

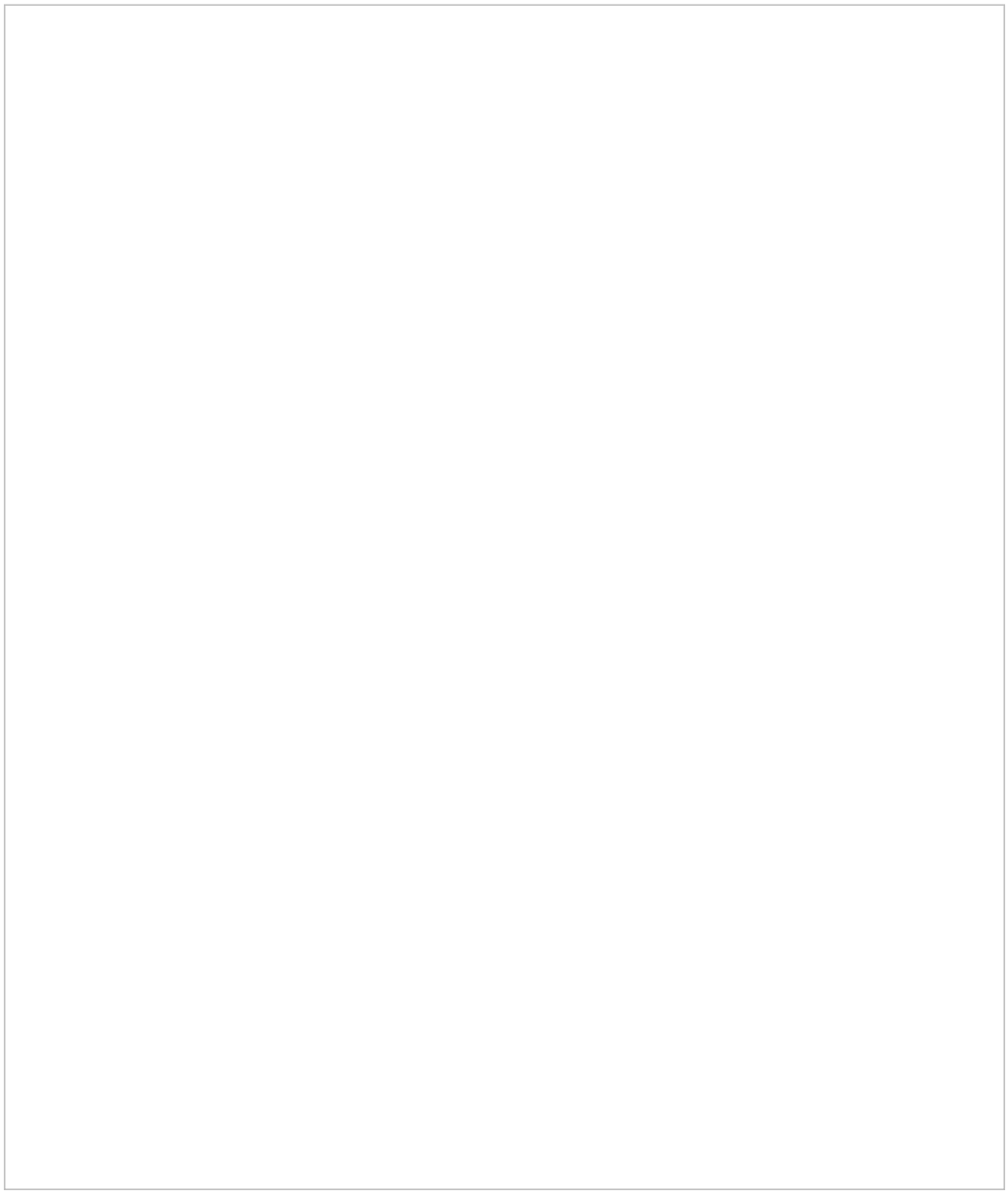
Step 9: Retrieve activities

In the fourth Fetch URL block, we will retrieve activities (e.g., your bot's answer) by using HTTP GET.

Set the following properties:

- **Title text** - Name the block (e.g., "Get Bot Answer")
- **Request type** - GET
- **URL to fetch** - https://directline.botframework.com/v3/directline/conversations/$(conversation.conversationId)/activities
- **Extra Headers** - Authorization: Bearer $(token)
- **URL parameters** - None

- **Content Type** - application/JSON
- **Body** - Empty
- **Username** - Leave blank
- **Password** - Leave blank
- **Initial path in the result JSON** - activities
- **Scenario variable prefix for JSON data** - azure_bot_answer
- **Use GetNext block to loop through data** - Select checkbox



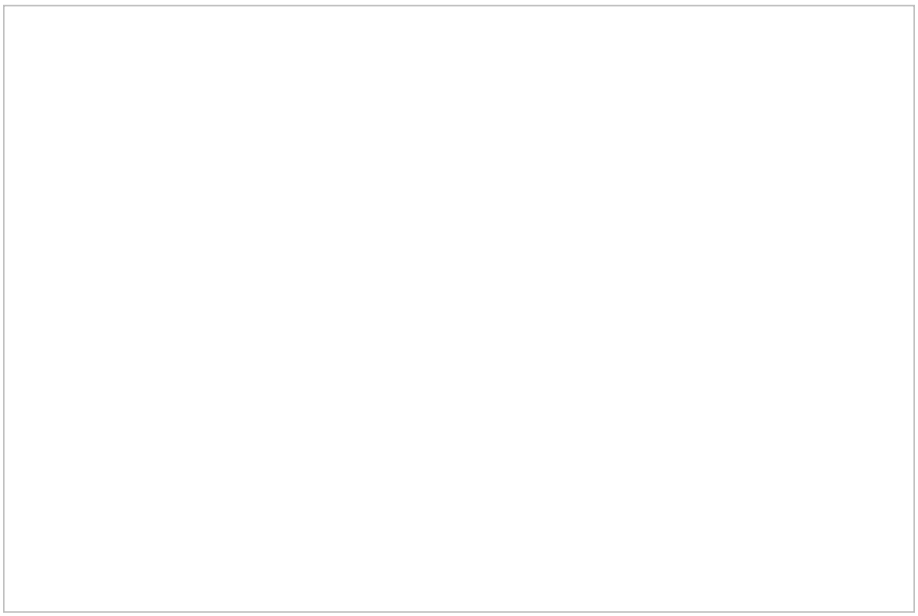
Fetch URL block being used to retrieve the bot's answer

Step 10: Set conditions for routing bot answers

In the If block and the Send Message+ block, we will specify that if the bot has an answer, then we will send that answer to the customer in a chat message using *$(azure_bot_answer.text)*.

Set the following properties:

- **Exit label** - Has an answer
- **Conditions**
    - Scenario variable (string)
    - azure_bot_answer.replyToId
    - is
    - =
    - $(message_id.id)

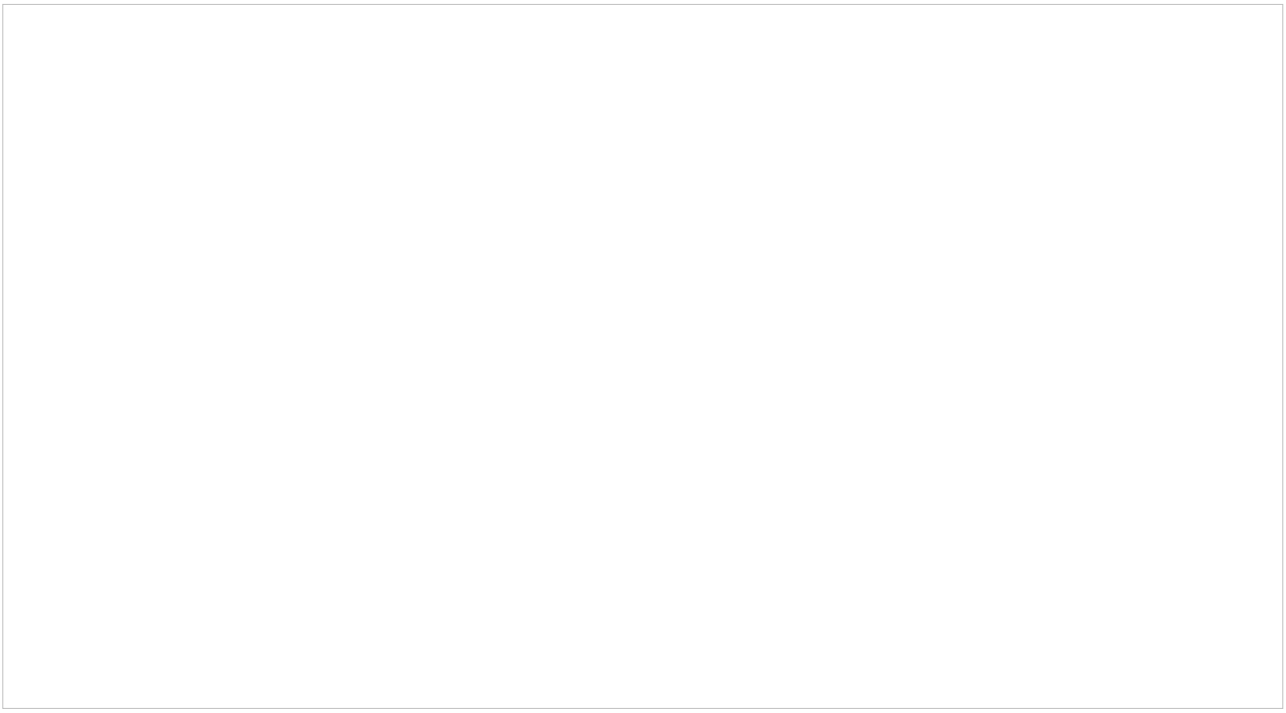


If block being used to set conditions to send the bot's answer to the customer, if there is one

Step 11: Pass along the bot's answer to the customer

Under the “Has an answer” exit, add a Send Message+ block and set the following properties:

- **Media type** - Chat
- **Message** - $(azure_bot_answer.text)



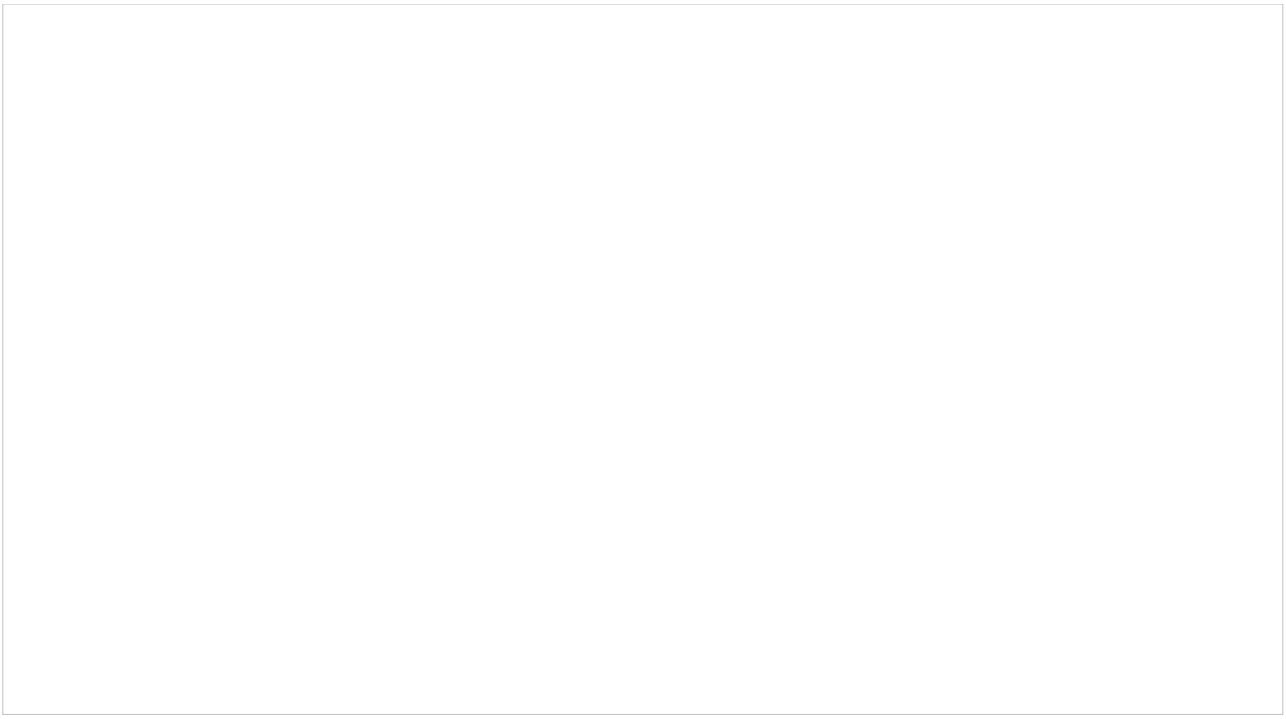
Send Message+ block being used to send the bot's answer

Step 12: Get the next bot answer

In the Get Next Record block, we will get the next bot answer, if there is one.

Set the following properties:

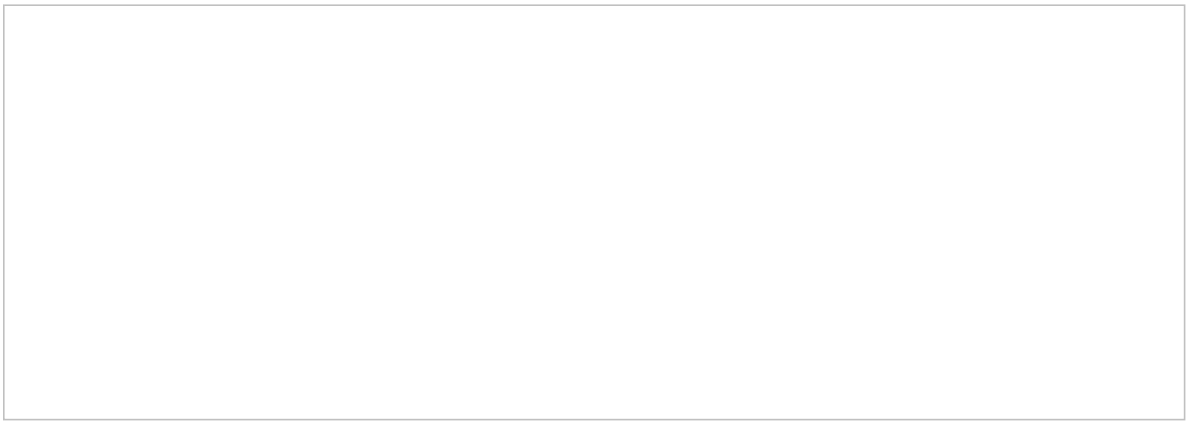
- **Title text** - Any name
- **Direction** - Next record
- **Recordset name** - azure_bot_answer



Get Next Record block being used to request the next bot answer

Step 13: Use Goto to request input again

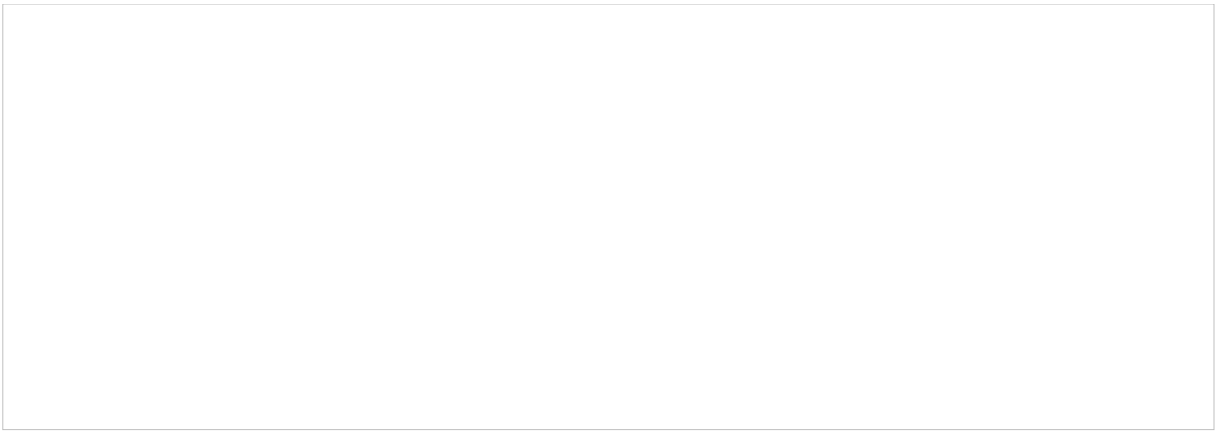
If there are no more answers from your bot, request input again by using the Goto block.



Goto block pointing to Request Input

Step 14: Keep checking for bot answers

Add another Goto block and point it to the If block. Doing so will create a loop that gets the bot's answer, sends it in chat, and checks for more bot answers until there are no more.



Goto block pointing to If block

Step 15: End the conversation

In the last Fetch URL block, we will end the conversation by sending a POST with the “endOfConversation” activity to the channel's messaging endpoint.

Set the following properties:

- **Title text** - Name the block (e.g., “End Conversation”)
- **Request type** - POST
- **URL to fetch** - https://directline.botframework.com/v3/directline/conversations/$(conversation.conversationId)/activities
- **Extra Headers** - Authorization: Bearer $(token)
- **URL parameters** - None
- **Content Type** - application/JSON
- **Body** - Specify the “endOfConversation” activity in the body:

```
{
   "type": "endOfConversation",
   "from": {
      "id": "user1"
   }
}
```

1. **Username** - Leave blank
2. **Password** - Leave blank
3. **Initial path in the result JSON** - Leave blank
4. **Scenario variable prefix for JSON data** - Leave blank

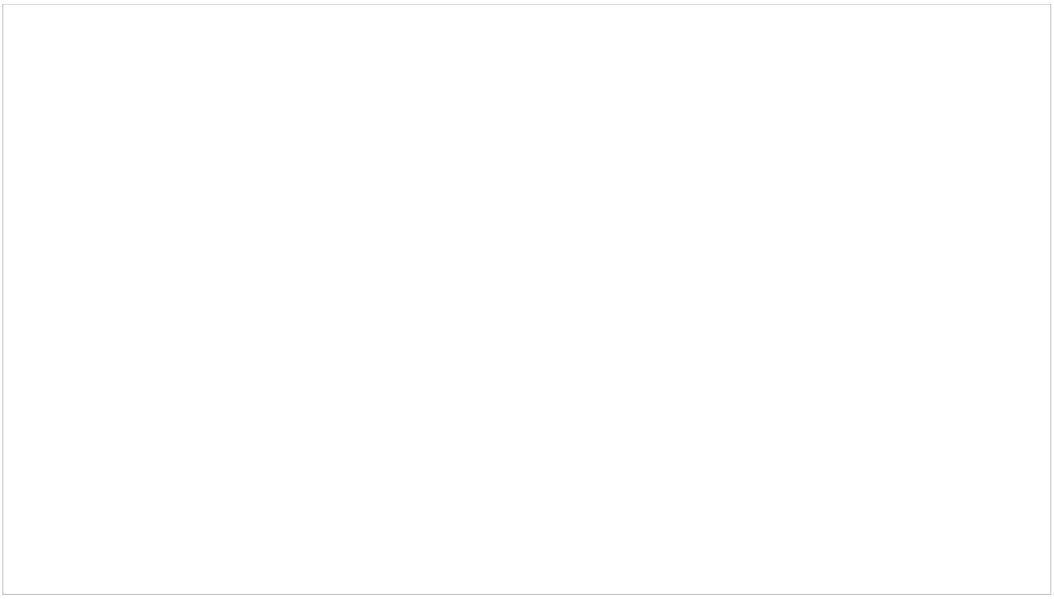
Fetch URL being used to end the conversation

Step 16: Save

Save the scenario.

Step 17: Configure your chat scenario entry to use the scenario

In the messaging/chat scenario entry you wish to use, make sure to set the correct scenario and service for your Web App Bot-enabled chat.



Messaging/Chat scenario entry properties

Your configuration is now complete. You can now launch a chat and try it.

How to Blacklist Specific Phone Numbers

Contact centers exist to make calls to and receive calls from customers. Unfortunately, contact centers sometimes receive inappropriate calls from customers. In order to protect your agents from abusive callers, Bright Pattern Contact Center software can be configured to blacklist specific phone numbers. Blacklisting or blocking a phone number means that any incoming call from that phone number is not allowed to be connected to the contact center.

This tutorial describes how to blacklist specific phone numbers.

You will learn how to:

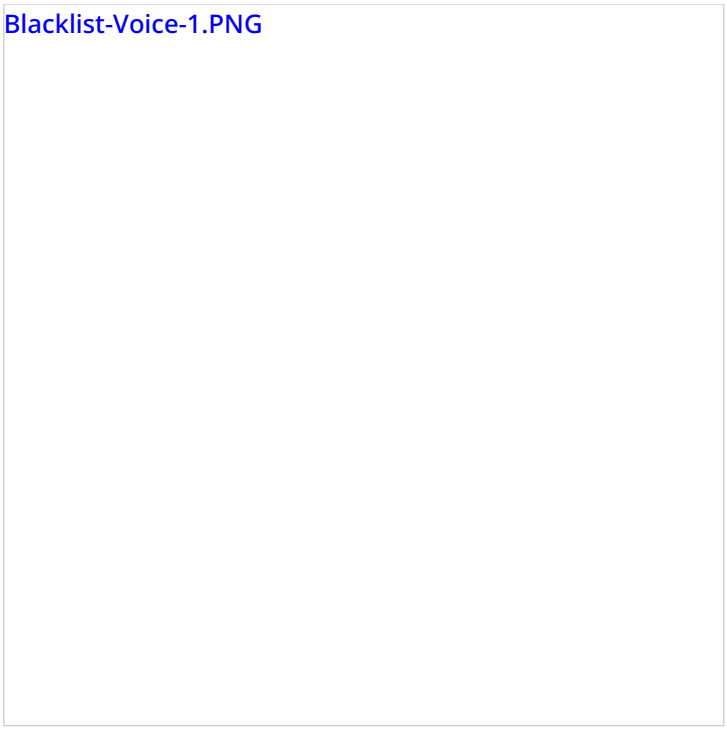
- Add a "blacklisted" flag to contact records
- Check if a caller's contact is blacklisted and reject calls by means of scenarios
- Add or edit blacklisted contacts in the Agent Desktop application

Procedure

1. Add a “blacklisted” flag to contact records

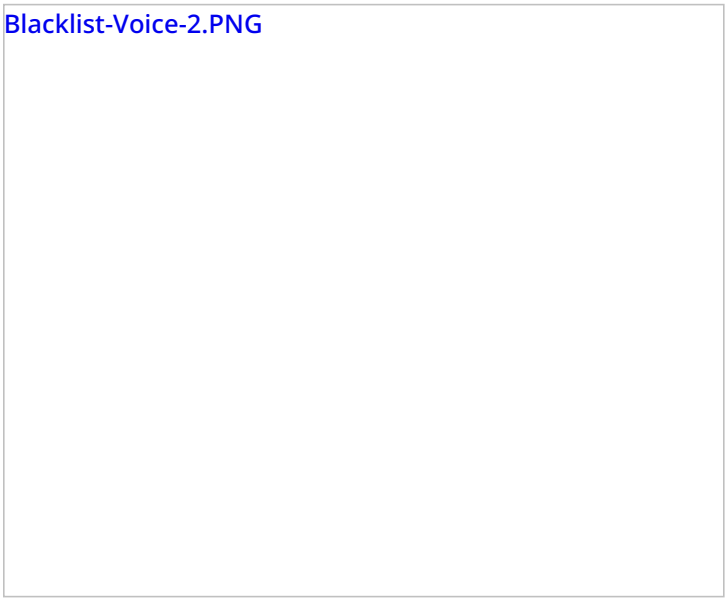
1. In the Contact Center Administrator application, section Case & Contact Management > Custom Fields > Contact, add a custom field (e.g., “blacklisted”). Click **Apply** to save.

Blacklist-Voice-1.PNG



2. Navigate to Case & Contact Management > Forms > Contact and open your default Contact Detail form; this will launch the Form Builder application.

Blacklist-Voice-2.PNG



3. In the Form Builder application, add a Text field to the Contact form, and then click the **edit** icon. When configuring the field, apply a Label, select the Editable option, and then map the custom contact field created in Step 1 to the Data field property. Click **OK** to save the field properties, click **Save** to save the form, and then close the application.

   Note: If you edit the system default Contact Detail form, when you save the form it creates a duplicate Contact Detail form. Rename the duplicate and select the option **Default form for this type**. Click **Apply** to save the changes.

Blacklist-Voice-3.PNG

2. Configure a scenario that identifies and rejects blacklisted callers

1. In the Contact Center Administrator application, navigate to *Scenarios* > *Voice*, and then open the voice scenario associated with your configured inbound voice service.

2. In the Scenario Builder application, add a Bright Pattern Search Object block to your scenario. Configure the block to include the following:
   - **Object type**: Contact
   - **Search**: Phone Equal ($item.from)
   - **Return fields**: The name of the custom data field that you created in the previous step, as it appears in the field properties (e.g., in the above example this would be "custom_blacklisted")
   - **Recordset name**: Any word or letters you wish to associate with this block (e.g., “RS”)

   Blacklist-Voice-4.PNG

3. Next, place an If block immediately after the Bright Pattern Search Object block you created in the previous step to your scenario. Add a branch, add a condition, and configure it as follows:
   - Scenario variable (string)
   - The variable returned by the previous block, prepended by the recordset name (e.g., “RS.custom_blacklisted”)
   - is
   - =
   - The symbol or string used in the contact form to specify the contact is blacklisted (e.g., 1)

   CCA-Blacklist-Scenario-5399webrtc.PNG

4. From the configured branch, you can route the blacklisted caller as you see fit. For example, you could add a Play Prompt block, configure it to play a message to the blacklisted caller, and then add an Exit block to terminate the call.

   When you are finished, click **Save** and exit the application.

   CCA-Blacklist-Scenario-1-5399webrtc.PNG

5. If necessary, repeat the four previous steps for your other inbound scenarios that you want to reject blacklisted numbers.

3. Add or edit blacklisted contacts in Agent Desktop

When you need to block a particular number, in the Agent Desktop application, sectionSearch & Preview Records, create a contact for the given caller or edit an existing contact. The field "Blacklisted" will be displayed on the Contact form. Enter a symbol or string that you used in the scenario (step 2.3 above) to specify that the contact is blacklisted (e.g., “1”) in this field, and then select **Save**.

You can repeat the above step at any time for any other numbers that need to be blacklisted.

Blacklist-Voice-6.PNG

How to Create a Voice Scenario That Distributes Surveys to a Percentage of Random Customers

Surveys are useful tools to get feedback from customers about your company, targeted topics, and so forth. After creating a survey, however, you may not want every single customer to have the option to take your survey. The following example voice scenario uses the random(max) function to distribute surveys to a set percentage of your callers at random.

Scenario Example

Click the following link to download an annotated version of this voice scenario example.
File:App How to Distribute Surveys to a Percentage of Random Customers.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

For general information about scenarios, refer to section Scenario Builder Overview.

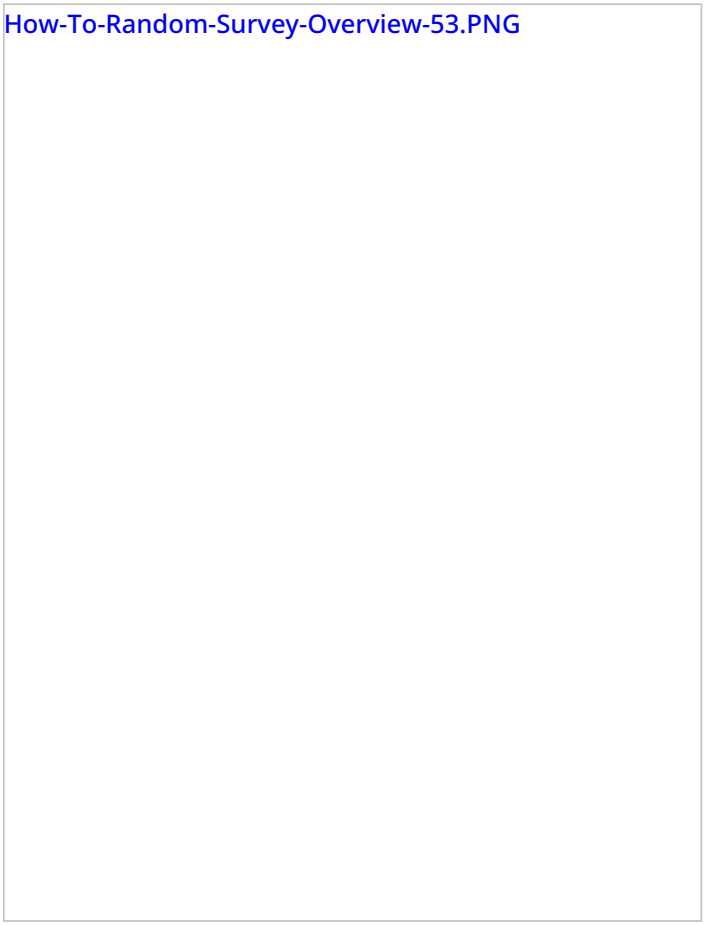
Scenario Flow

What follows is a simple sequence of actions that allows you to send a survey to 30% of random callers if the Connect Call block's Target Disconnected conditional exit is reached (i.e., the call ends normally). The survey is configured in a separate scenario that is triggered when a Start Another Scenario block is reached.

Please note this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.

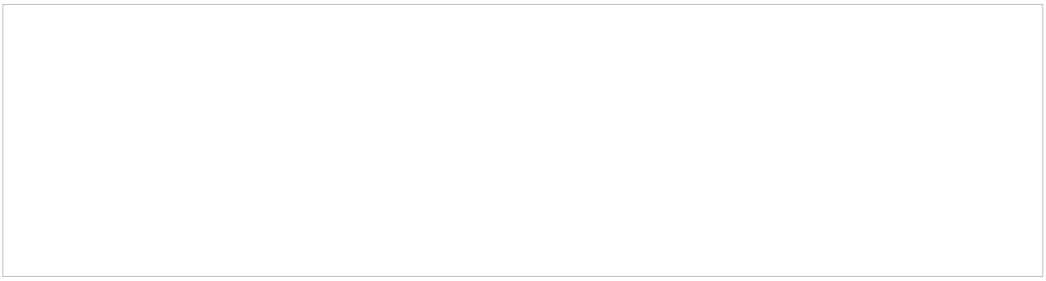
Scenario Overview

The diagram shown illustrates what the complete scenario looks like when designed in the Scenario Builder application.



Action 1: Begin with a Set Variable block that invokes random(max)

The first block in our scenario is Set Variable; we will use this block to invoke the *random(max)* function. In order to do this, we define the variable name (i.e., "sendsurvey") and value (i.e., "=random(100)").



The Set Variable block with random(max)

There are two important points to note about this:

1. The random(max) function works by selecting a random integer between 0 and one less than whatever number you configure as "max". For example, *=random(4)* will randomly return the values 0, 1, 2, and 3 (i.e., a total of four integers).
2. When configuring the "max" value, make sure this number is easy for you to work with; that is, it is half of the formula we will use to define the survey percentage. Our example sets "max" to 100.

As a reminder, in order to use functions, you must include the **=** symbol (i.e., *=random(100)*).

Action 2: Create a basic scenario

Next, in order for your voice scenario to work, you will need to include the Find Agent block and Connect Call block. These two blocks are the most important for any scenario where agents interact with customers directly.

For more information on using Find Agent and Connect Call scenario blocks, see How to Create a Basic Scenario.

Action 3: Configure an If block with the desired percentage of survey distribution

After configuring the Find Agent/Connect Call block combination, we will add an If block to the Connect Call block's Target Disconnected conditional exit. Note that this exit is reached when a call ends normally (i.e., the caller hangs up).

In the If block, we will do the following:

- Create a branch
- Add the condition being sure to select the following criteria:
  - **Scenario variable (number)**
  - **sendsurvey** (the variable we created with the Set Variable block)
  - **is**
  - Greater than symbol (**>**)
  - The desired number

We want our survey to go to 30% of callers randomly, so remembering our max number is 100, we will make the If block consider numbers greater than 70 only.

The If block is where the percentage is defined

Action 4: Use the Start Another Scenario block to initiate the survey

If the conditions of the If block are met, we want this to trigger a survey for the customer. In order to do this, we will add a Start Another Scenario block after the If block, which launches our survey scenario.

For more information on how to configure a survey scenario, see How to Create a Voice Scenario Survey.

The survey will be initiated if the Target Disconnected conditional exit condition is met

Suggested Reading

In order to better understand the blocks and variables used in this scenario, we recommend reading the following articles:

- Built-In Functions
- Set Variable
- If
- How to Create a Basic Scenario
- How to Create a Voice Scenario Survey

And finally, don't forget to save your work early and often!

How to Configure Last-Agent Routing Using the Internal Database

Sometimes, a customer will need to make several calls to your contact center. In the interest of good service and continuity, you can configure last-agent routing, which makes it possible for the customer to continue speaking with the same agent (i.e., if they are available) when they call back.

This tutorial describes how to route a returning, identified caller to the last agent the caller previously spoke with.
Note: This scenario is designed to work with the internal database.

You will learn how to:

- Configure the correct identification settings for your contact center
- Identify the returning caller by finding their contact information in the internal database
- Locate the agent that last spoke with the returning caller, then route the call to this agent

As a reminder, please save your scenarios early and often!

Scenario Example

Click the following link to download an annotated version of this voice scenario example.

Media:App_Last-Agent_Routing.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

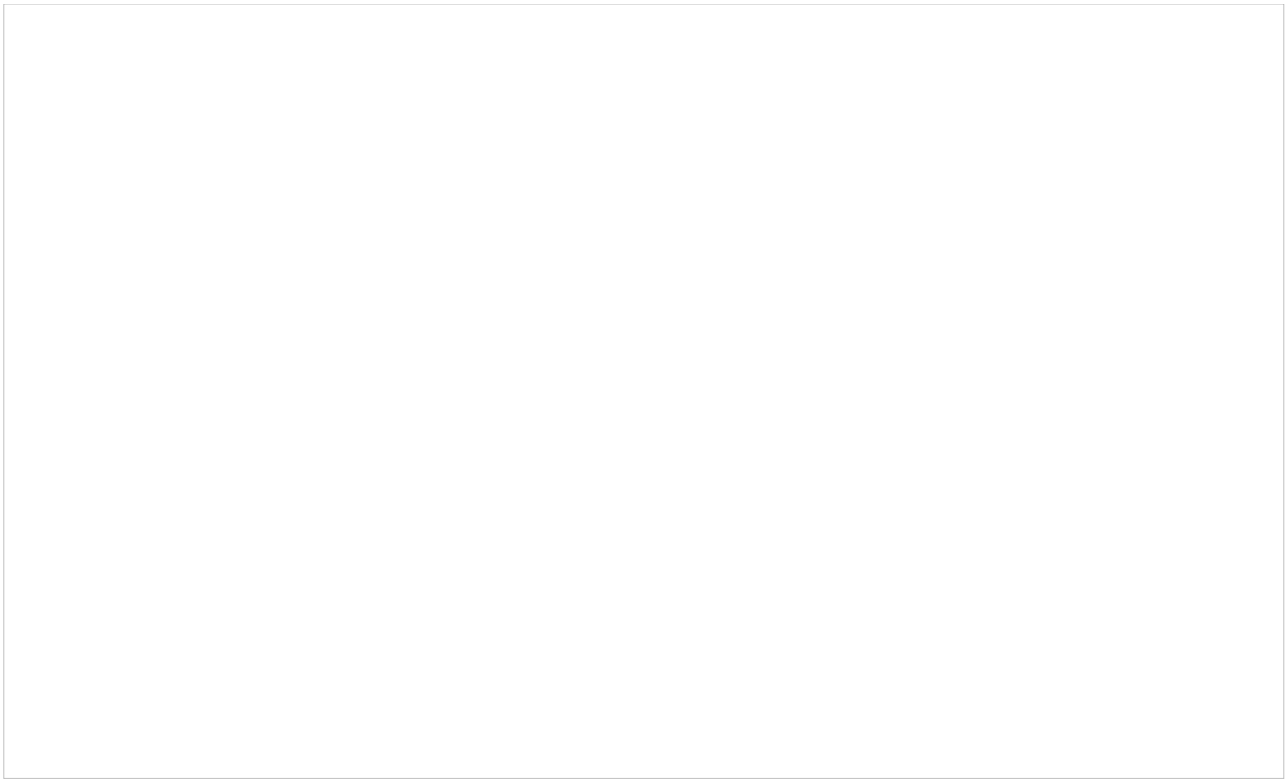
For general information about scenarios, refer to section Scenario Builder Overview.

As a reminder, this scenario is an example for testing purposes only and is NOT intended for production use.

Procedure

1. Configure your contact center's identification settings

1. In the Contact Center Administrator application, go to section Call Center Configuration > Identification.



Call Center Configuration > Identification

2. If you have not already done so, enable the setting Enable Service continuation for up to __ hours, and define the appropriate number of hours.

Enable service continuation

3. Next, if you have not already done so, enable the setting Use Internal Contact Database. This specifies the system will use the internal database only. Optionally, you may choose to enable the setting Always create contact in Internal Database. When enabled, this setting automatically creates a contact in your internal database for inbound interactions.

   Note: If you are using an external database, the setting *Always create contact in Internal Database* allows you to define Salesforce.com or Zendesk integration accounts.

4. When you are finished, click **Apply** to save your settings.

2. Greet the caller

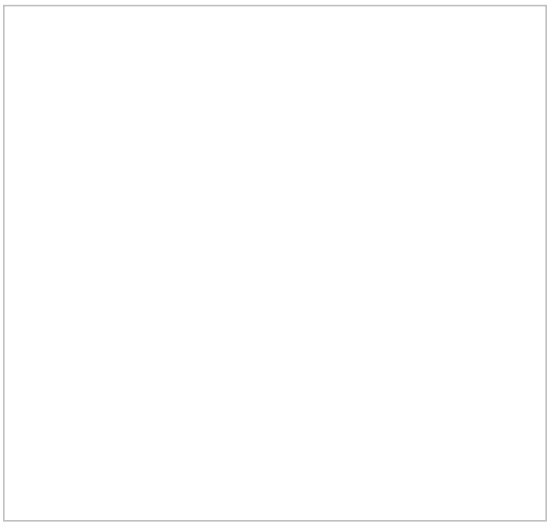
1. Next, go to section Scenario > Voice, and click + to create a new voice scenario; this will launch the Scenario

Builder application.

2. The first block we add to this scenario is the Play Prompt block. The block will not only greet the caller, but it will inform them they must wait until an appropriate agent is located.

   Adding this block at the beginning of the scenario is especially useful for allowing the system time to find and retrieve the caller's contact information and/or the previous agent the caller spoke with.

   Note that contacts in external databases might require slightly more wait time. In this instance, you can add an additional audio treatment to fill any time gaps. For example, you may add another *Play Prompt* block with a "please wait" message and/or music.



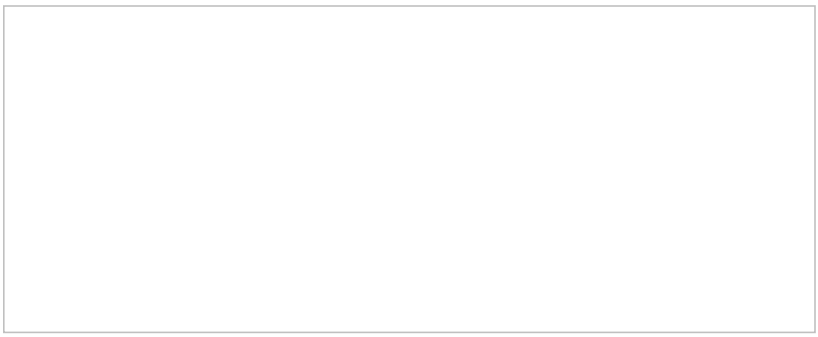
Greet your caller

3. Identify the returning caller

1. The next block you will add to your scenario is the Identify Contact block; it allows you to search an internal or external database for a matching saved contact.

2. In our example, we will search for a contact with a phone number matching the phone being used for the incoming call; this is passed along the system using the variable $(item.from).

3. **Note**: If the incoming caller's phone number does not match a saved contact, you can add Goto blocks on the conditional exits that redirect the caller to another agent.



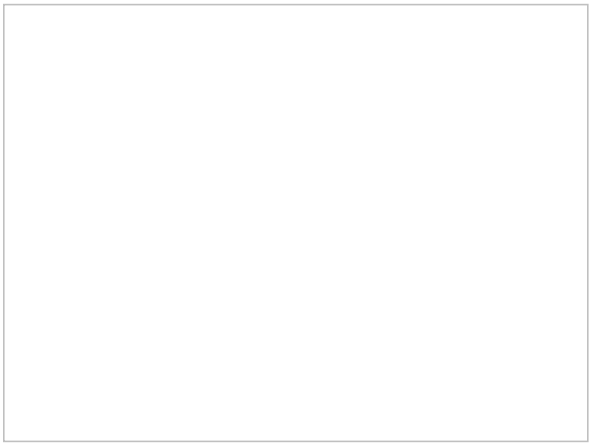
Identify Contact can search in internal or external databases

3. Identify the previous agent

1. Next, you will add the block Get User Configuration. When configured, it will identify the last agent the returning caller spoke with.

2. In order to do this, configure the Find user by setting to **User ID**, and the Value field to contain the variable $(item.continuationUserId). This variable finds the user ID of the agent the caller previously spoke with.

3. Then, for the User properties to return setting, we will set it to return the found **User ID**, and then name the return results variable *UserID* (i.e., you can put any text in this field). This variable will be used in the Find Agent block.



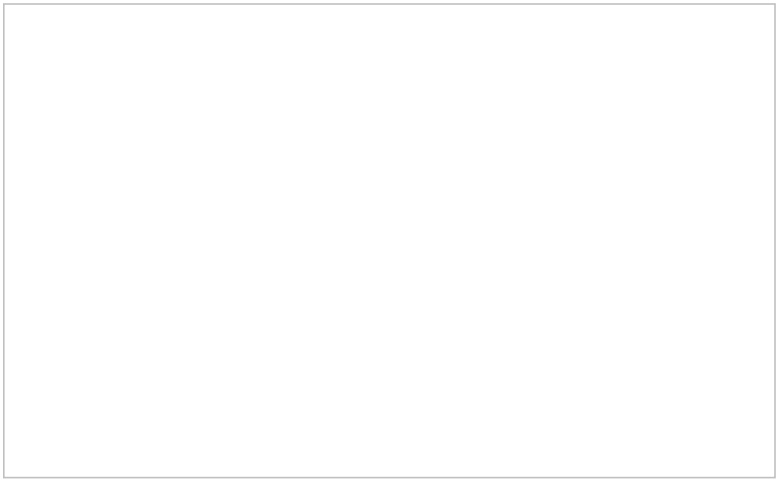
Use the variable $(item.continuationUserId)

4. Route the returning caller to the previous agent

1. The next block to add is the Find Agent block. In the Agent skills required setting, you will select the **Specific agent** wait condition, then add the variable *UserId* from the Get User Configuration block. Note that this condition is available for the first interval **only** (e.g., 0 - 1 sec). This configuration ensures that if both the saved contact and the previous agent are found and available, they will be connected first.

Route the call to the specific agent

2. In the event that neither the contact nor the agent is found/available, you will need to define an additional *Agent skills required* option (e.g., Services >= 100). This ensures the caller will be routed to the next most appropriate and available agent.

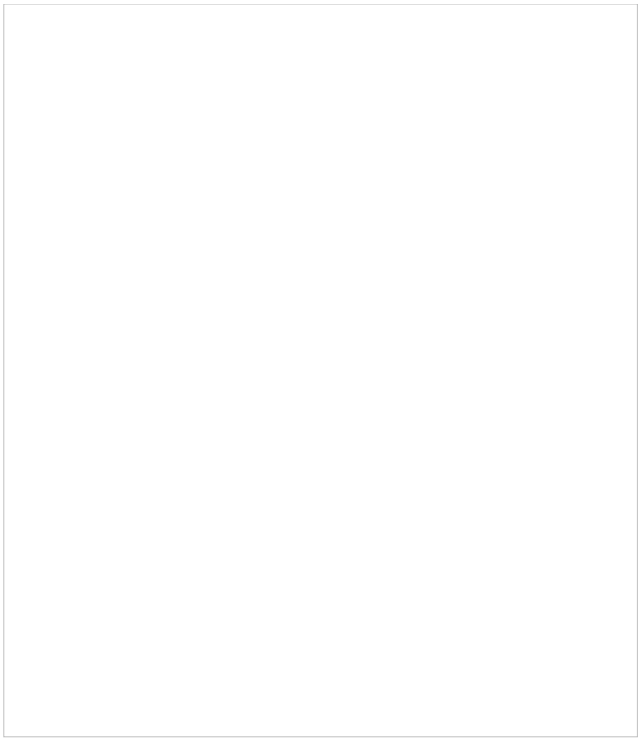


Ensure your caller is routed to an available agent

5. Connect the caller to the agent

1. Finally, once an available agent is found with the Find Agent block, the caller will be connected to them with the Connect Call block; note that these two blocks should be arranged in consecutive order withno other scenario blocks between them. As a reminder, please define all conditional exits.

2. After the Connect Call block, you may add an Exit block to end this scenario.



Scenario overview

How to Route Callers to the Last Agent and Provide a Voicemail Option

So your customer is calling again and wants to talk to the last known agent who helped her. How do you connect them?

This article will show you how to create a general voice scenario that

- Finds the last known agent (also called same agent routing)
- Connects the call to that agent or presents an IVR menu with the option to either leave a voicemail or look for the next available agent

Prerequisites

Before proceeding, we recommend that you download and import our example scenario template: File:App FindLastAgentVM (2).zip

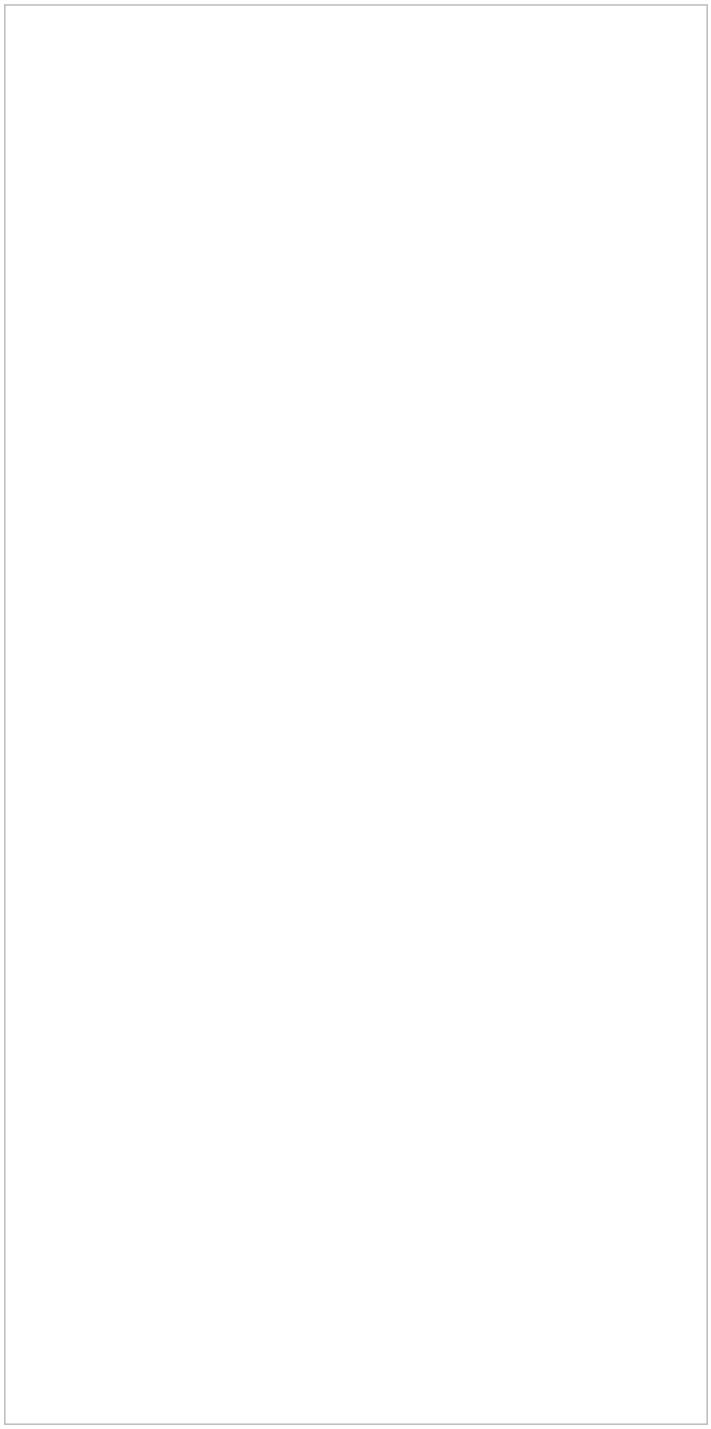
For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

Scenario Flow

Scenarios are made up of *scenario blocks* that specify how interactions are processed. Detailed descriptions of Bright Pattern scenario blocks are available in this guide's *Scenario Block Definitions* section.

The following sequence of actions illustrates one way to organize scenario blocks to route a customer to either the last known agent, an IVR menu with the option to leave a voicemail for that agent, or to the next available agent.

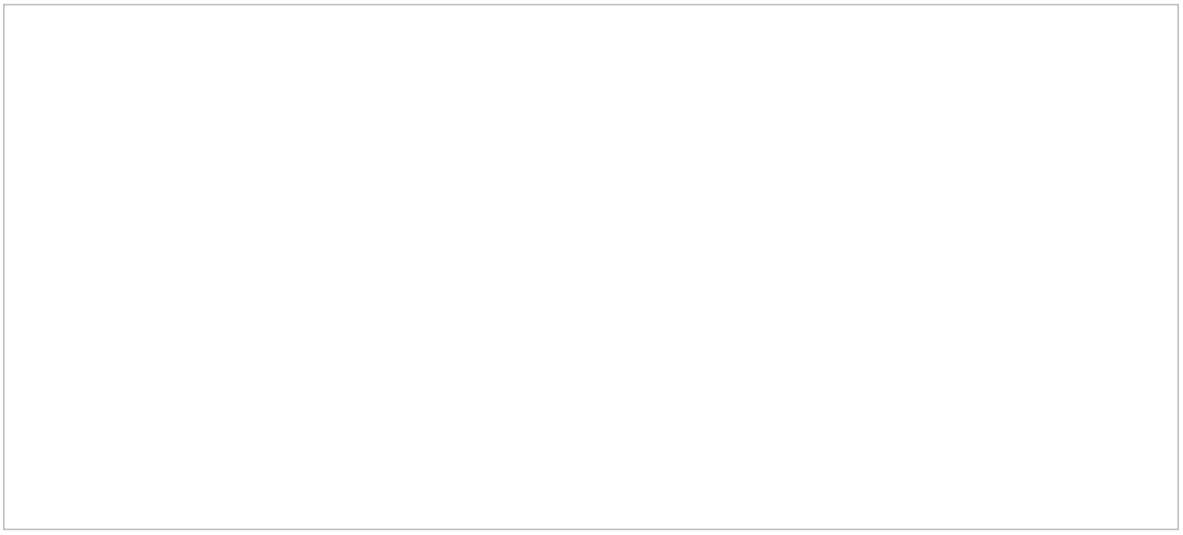
Please note this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.



Example scenario in Scenario Builder

Action 1: "Get user configuration" looks up the last known agent

The Get user configuration block finds a specific contact center user and returns the user ID of the last known agent, using variable "lastagent."



The requested user properties are returned

Action 2: Exception Handler's "Try" exit tells the scenario what might generate an exception

The Exception Handler block has two conditional exits: Try and Catch.

Under Exception Handler's "Try" exit, you should enter the sequence of blocks that you predict might generate an exception, block error, or disconnect. In this example, we did so in the following order.

Find Agent

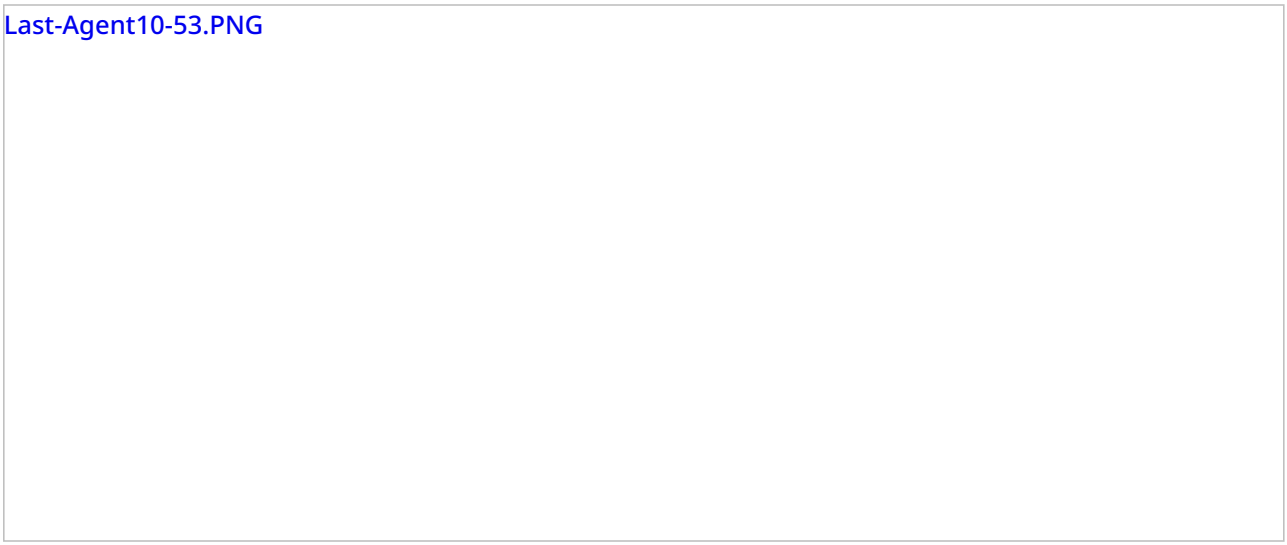
Find Agent looks for the last known agent.

Last-Agent-2-53.PNG

In Find Agent properties, you must also use a "Specific agent" condition to specify which agent needs to be found.

1. Under "Agent skills required," click **add Interval**.
2. Under "Conditions," click **add**.
3. From the drop-down menu, select **Specific agent**.

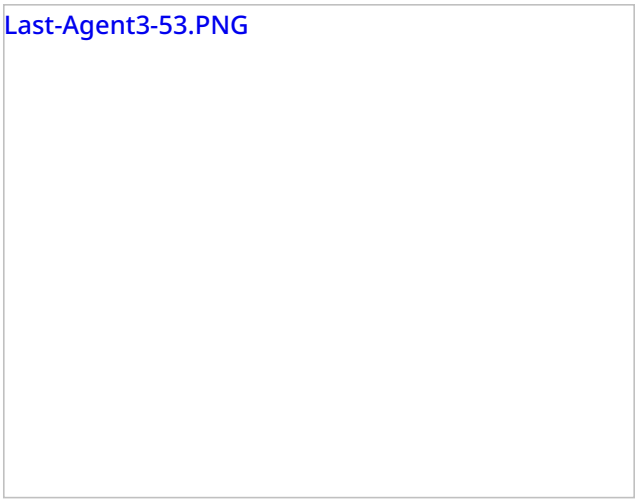
Last-Agent10-53.PNG



4. In the variable value field, enter variable **$(item.continuationUserId)**. This variable is used to pull the last known agent's user ID when service continuation is enabled for your contact center. See the Final Step of this procedure for more information.

Play Prompt

Play Prompt plays the voicemail option voice segment to the caller (e.g., "To leave a voicemail press 1, to speak with an agent now, press 2").



Menu

Menu specifies the two available IVR menu options as given in the Play Prompt block (i.e., 1 for voicemail, and 2 for finding an agent).

Voicemail

Voicemail tells the system how to encode the voicemail recording, who to deliver it to, and so forth.

Last-Agent6-53.PNG

Goto

Goto jumps to "Find Agent."

Last-Agent7-53.PNG

Action 3: The Exception Handler's "Catch" exit tells the scenario which blocks to use if an exception occurs

After defining the possible blocks that might start an exception, you work on the Catch exit.

Under Exception Handler's "Catch" exit, you should enter the sequence of blocks that you *want the scenario to execute* if an exception, block error, or disconnect occurs during the Try conditional exit. In this example, we did so in the following order.

Connect Call

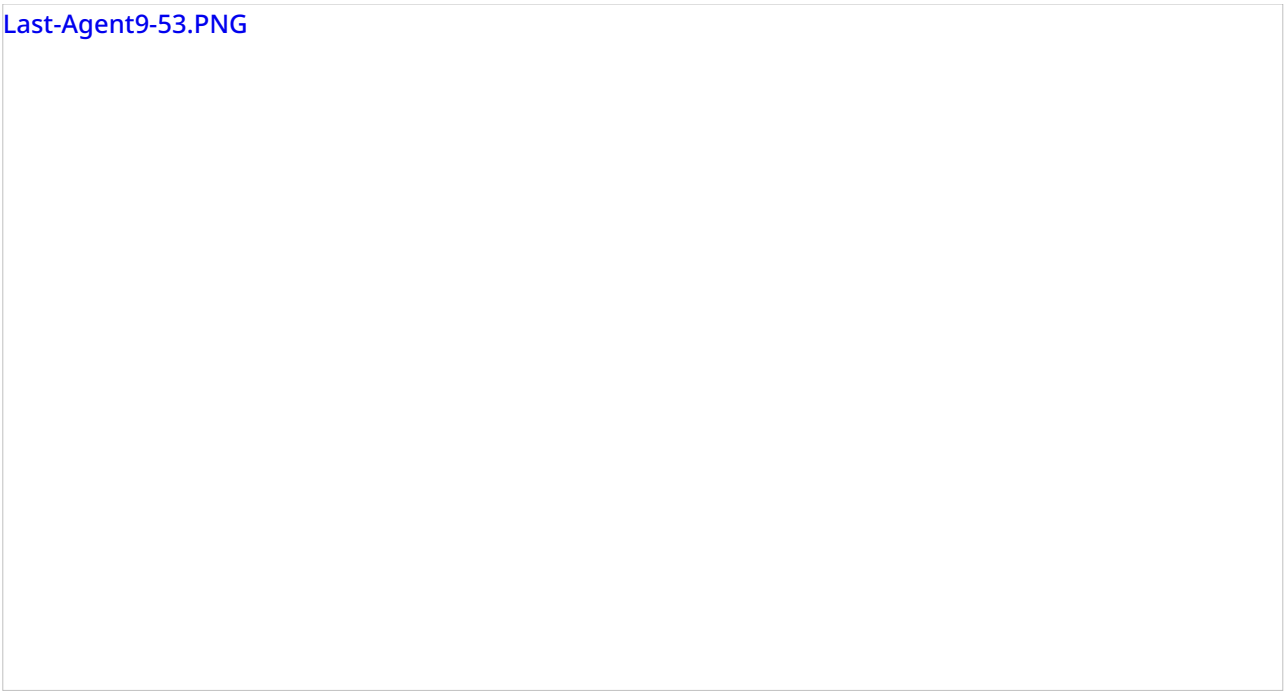
Connect Call connects the customer to the last agent.

Last-Agent8-53.PNG

Find Agent

Find Agent finds the next available skilled agent (not the last known agent). There are no special conditions to add in this Find Agent block.

Last-Agent9-53.PNG



Action 4: "Exit" ends the scenario

The Exit block completes the scenario. Without it, the scenario will loop through this configured flow until the customer ends the call.

Action 5: Save your scenario

Be sure to save the scenario with a new name. Closing the browser tab or window will cancel out your work.

Final Step: Enable service continuation

Now that you have completed your scenario, you need to enable service continuation. This allows your contact center system to return an identified contact (i.e., your caller) to the same agent who last helped the caller.

1. In the Contact Center Administrator application, go to *Call Center configuration* > *Identification*.
2. Enable **Enable Service continuation for up to __ hours** and specify the number of hours.

Call Center configuration > Identification

For more information, see the Contact Center Administrator Guide, section Identification.

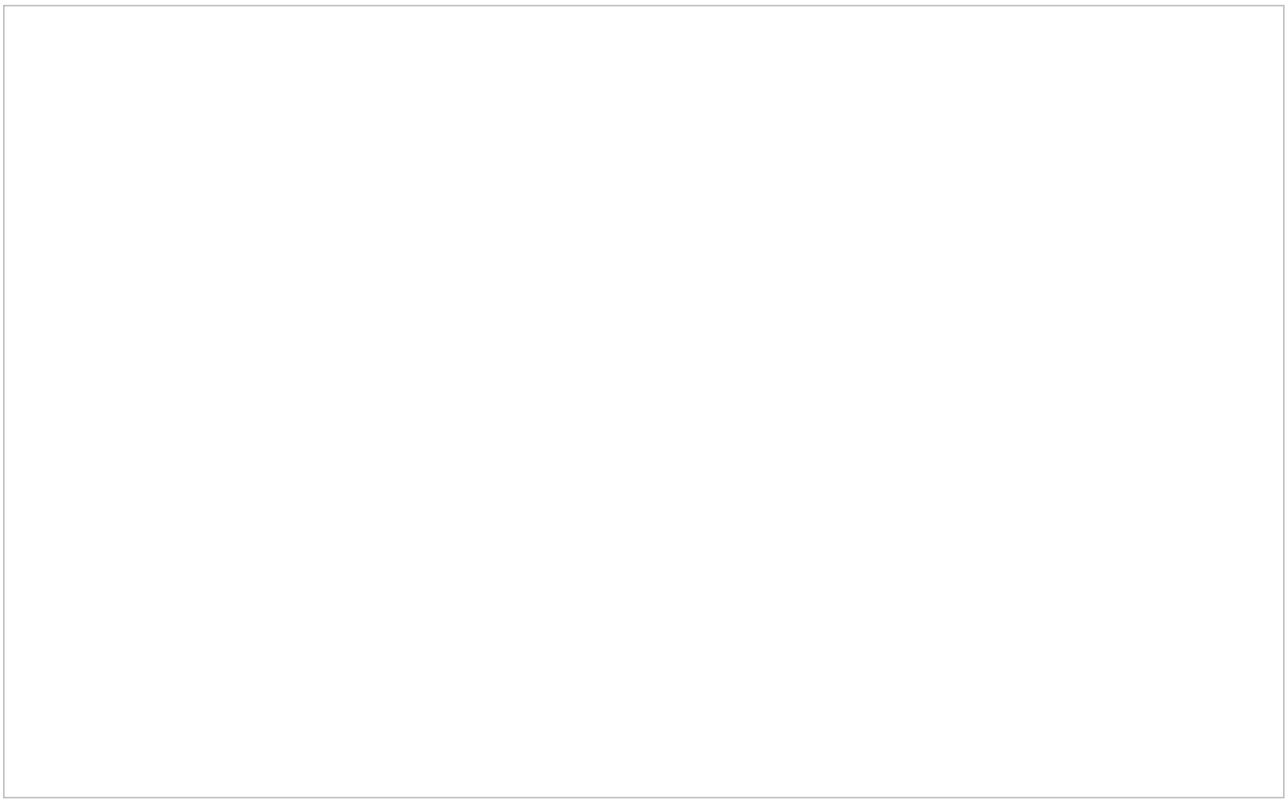
Related Information

Note: It is possible to route additional interaction types from a customer to an agent during an active interaction (e.g., a caller sends an SMS to an agent during the same interaction) using the Find Agent block's wait condition Agent currently handling this interaction.

Sending Automatic Email Replies to Customers Who Use the "Leave a Message" Chat Form

If your contact center offers chat services within a set of operating hours, theLeave a Message Form option allows customers to use the chat widget to send you email messages outside of these hours of operation.

Using the Scenario Builder application, you can configure the system to send automatic email replies to customers who send after-hours messages. This ensures you're providing great customer service even when your contact center is closed.



Configuring the Leave a Message chat form

Prerequisites

Note that this scenario requires you to have configured chat and email services, configured chat and email scenario entries, as well as a working SMTP configuration.

Because a key part of this example is based on hours of operation, it is helpful to understand how to configure them in your contact center. And finally, you may find it helpful to review the variable $(item.externalChatData) as it refers specifically to fields in the chat widget.

For more information about configuring the chat widget, see *Administration Tutorials*, section Chat, as well as the *Chat Widget Configuration Guide*.

Scenario Example

Click the following link to download an annotated version of this chat scenario example.
File:App After-Hours Send an Email Reply to a Chat Customer.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

For general information about scenarios, refer to section Scenario Builder Overview.

Example Flow

This example illustrates how to configure the Leave a Message form in the Chat Widget Configuration application, focusing on field variables. Then, in the Scenario Builder application, we configure an If block to perform an after-hours check. If messages are received outside of the configured hours of operation, we use the EMail block with the chat widget's field variables to send email replies to customers.

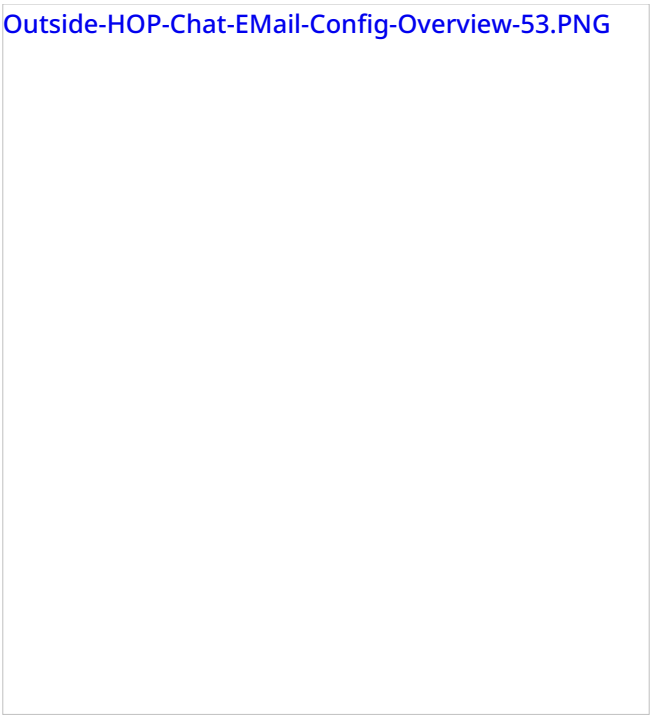
Designer's note: The same scenario blocks can be used to route the chat message directly to your contact center's email address, for troubleshooting purposes.

Note that this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.

Scenario Overview

The diagram shown illustrates what the complete scenario looks like when designed in the Scenario Builder application.

Outside-HOP-Chat-EMail-Config-Overview-53.PNG



Action 1: Configure the "Leave a Message" Chat Form in the Chat Widget Configuration Application

In order for our scenario to work, we must configure the Leave a Message form in the Chat Widget Configuration application first. To launch the application, in the Contact Center Administrator application, go to section Scenario Entries > Messaging/Chat > Chat Widget tab, and click **add** under Chat Initiation via Contact Tabs.

In the application, configure the fields you want your *Leave a Message* form to have. Note that the Email to send form to field should contain the email address you want to receive the results of these forms.

When adding fields to your form, note that the *name* can be added to the end of the variable $(item.externalChatData) as a way to reference back to a specific field (e.g., the field named *subject* can be added like so: *$(item.externalChatData.subject)*).

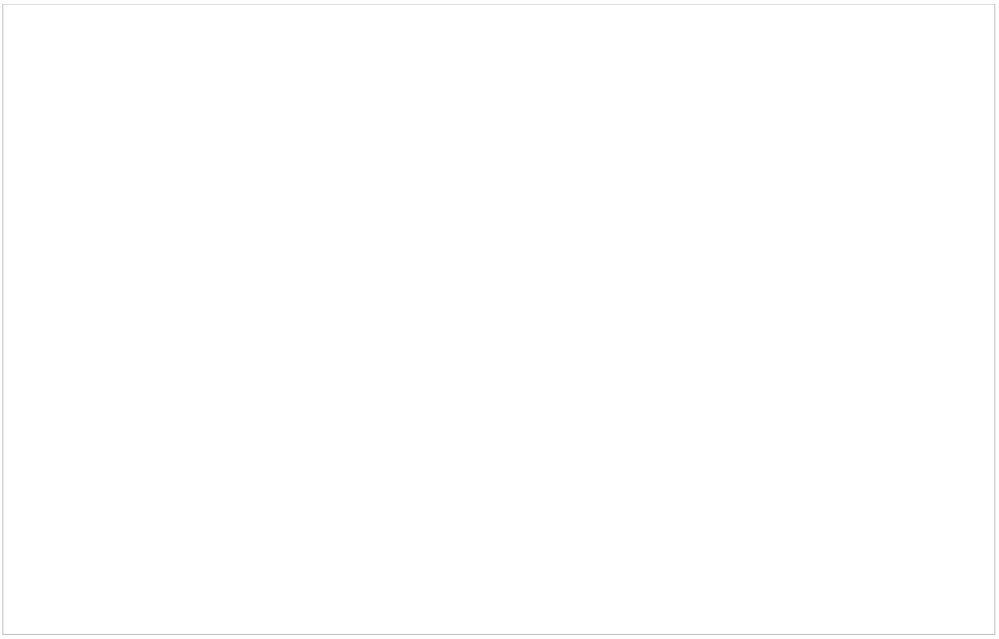
Field names are used with the variable $(item.externalChatData), so make sure you keep track of them

Action 2: Configure an If Block to Perform an After-Hours Check

After configuring the chat form, in the Scenario Builder application, we want the system to check if the chat is received outside of the service's hours of operation. To do this, add an If block, add a new branch, enter an exit label, then add a condition that specifically looks at hours of operation. The condition is configured like so:

- "The current date and time"
- "is not"
- "Scenario default hours of operation"

Additionally, note that you may configure specific hours of operation entries in the final field.



Configure an after-hours check using the If block

Action 3: Add the EMail Block to the If Block's After-Hours Check Branch

If the chat is received outside of the specified hours of operation, we want the system to send out an automatic email reply; this is done by adding the EMail block to the after-hours branch we created with the If block.

When configuring the EMail block, and to ensure the email populates with information the customer entered in the widget, use the variable $(item.externalChatData) where applicable.

The field names from the widget should be added to the end of the variable to pull the correct information, for example, if the widget's email field is named *email*, the variable would be *$(item.externalChatData.email)*.

Populate the EMail block with the variable $(item.externalChatData) where applicable

Action 4: Configure Find Agent / Connect Chat

If the chat is received during normal hours of operation, the customer will need to be connected to an agent. To do this, configure your scenario with the two most basic chat scenario blocks: Find Agent and Connect Chat. *Find Agent* is where the system finds the most appropriate agent to route the chat to; *Connect Chat* opens the chat channel between the found agent and the customer. All chat scenarios will contain the Find Agent block and Connect Chat block; note that they should be arranged in consecutive order.

For more information about these blocks, see How to Create a Basic Scenario. As a reminder, define all conditional exits.

Skill-Based Call Routing with an Auto Attendant Choice

In the Bright Pattern Contact Center realm, *skills* are the various abilities of users that can be defined in the Contact Center Administrator application. Skills can be defined as any topic or language pertinent to your contact center (e.g., accounting, Arabic, IT, etc.) Skills are rated in section Skill Levels per user from 0 to 100 (i.e., 0 being the lowest skill possible, and 100 being the highest skill possible). Additionally, users can be skilled for a service or campaign.

Through scenarios, it is possible to route interactions to higher-skilled agents before lower-skilled ones. This lets your customers interact with your most knowledgeable agents, which provides the best customer service experience. Additionally, skill-based routing is great to use when training new, lower-skilled agents; this provides them the opportunity to learn while preventing them from being overwhelmed with interactions.

This scenario details how to configure basic skill-based routing with an option for customers to enter in an extension directly (i.e., "auto attendant"). The skills used in this example are Auxiliary Skills but they may be changed to Language Skills (e.g., your contact center provides services in English, French, and Spanish).

Scenario Example

Click the following link to download an annotated version of this chat scenario example.
File:App Skill-Based Call Routing + Auto Attendant.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

For general information about scenarios, refer to section Scenario Builder Overview.

Scenario Flow

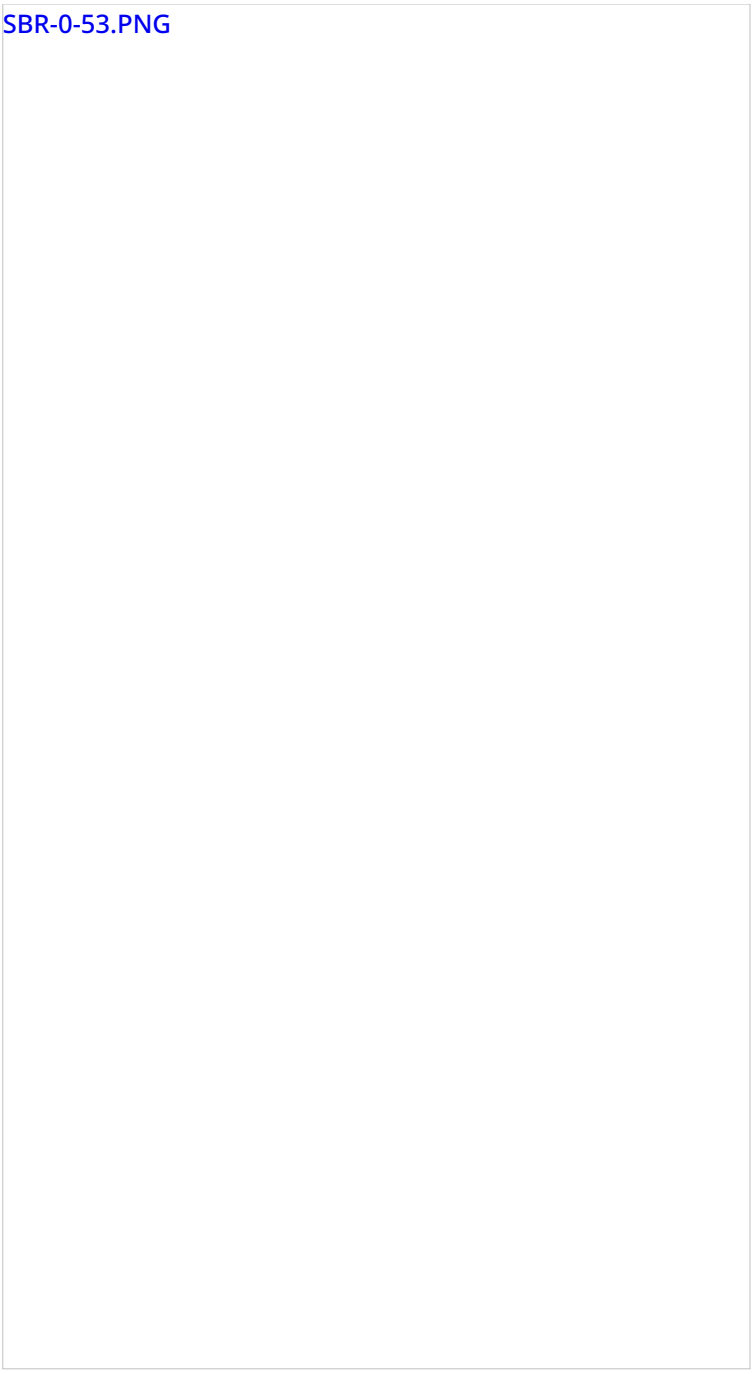
This voice scenario uses a menu to route customers to skilled agents and includes an option to dial the extension of a specific agent. This is accomplished by using the following blocks: Menu, Request Skill or Service, Collect Digits, Set Variable, Find Agent, and two instances of Connect Call.

Note that this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.

Scenario Overview

The diagram shown illustrates what the complete scenario looks like when designed in the Scenario Builder application.

SBR-0-53.PNG



Action 1: Create a Menu with Skill-Based Choices and an Auto Attendant Choice

In this scenario, the Menu block allows callers to select a menu choice that routes them to either a skilled agent or allows them to enter the extension of a specific agent directly.

When configuring the Menu block, we create a prompt to play, then configure our valid menu choices. In this example call center, the skills are "Customer Service," "Accounting," and "Shipping and Logistics," so we create valid choices in the block for each of these. Additionally, we create the choice for the "Auto Attendant" (i.e., the caller can enter an extension).

Menu block configuration

Action 1a: For Skill-Based Menu Choices, Use the Request Skill or Service Block

Next, for each skill-based choice we added in the Menu block, we add the Request Skill or Service block and configure the corresponding skill. This is done by selecting the skill group from the first drop-down menu (i.e., auxiliary, language, etc.), then the specific skill from the skill group in the next drop-down menu.

Note that the Request Skill or Service block acts as a connector between the Menu block and the Find Agent block, where the skill-based routing happens.

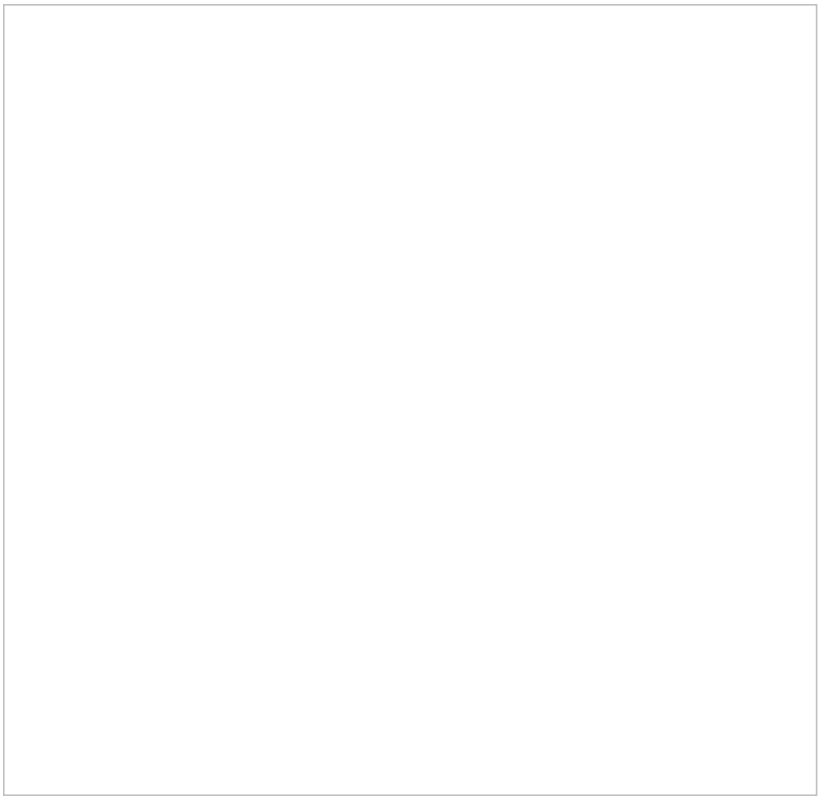
Request Skill or Service block configuration

Action 2: Begin Configuring Auto Attendant with Collect Digits

For the Menu block choice "Auto Attendant," we allow customers to connect to a specific agent via the agent's extension number. This is accomplished by using the Collect Digits block, the Set Variable block, and a Connect Call block.

All extensions in our example call center begin with 10 (e.g., 1003), so, rather than have the customer enter in the full four-digit extension, we configure a prompt to play that tells them to enter the last two digits of the extension.

The Collect Digits block records the two digits the customer enters and stores them in the variable configured in the field Name of the variable to store the result; in our example, we call this variable *extension*.



Collect Digits block configuration

Action 2a: Use Set Variable to Alter Information from Collect Digits

Next, the variable *extension* we created in the Collect Digits block is passed to the Set Variable block; this is where the first two digits of the actual extension are added back on to the variable. This is done by appending the number "10" before the begining of the variable like so: *10$(extension)*

For example, a customer enters in *39* in the Collect Digits block, which, when passed through Set Variable, becomes *1039*, the actual extension.

Set Variable configuration

Action 2b: Have Connect Call Receive the Altered Information from Set Variable

Finally, the variable *$(extension)* we created in the Set Variable block is passed to the following Connect Call block and entered in the Override Destination field. This ensures the call will go to the specific extension rather than any available agent.

Note that in this branch of the scenario (i.e., the "Auto Attendant" menu choice), if the customer does not enter an extension or the system does not recognize it, we use Goto blocks to route the caller to the Find Agent block.

Connect Call block configuration

Action 3: Configure Skill-Based Routing in Find Agent

If the caller selected a skill-based choice from the menu (i.e., they did not select the "Auto Attendant" choice), we will need the call to go to the most appropriately skilled agent.

To accomplish this, the Request Skill or Service blocks we configured for each Menu block choice are configured as Wait Conditions in the Find Agent block. The *Wait Condition* allows the system to find the most appropriately skilled agent for the interaction; however, if the skill condition is not met within the Wait time, the caller will be routed to the next skill group, and ultimately, any available agent.

Find Agent block configuration

Action 4: Connect the Call

After an agent has been found, the call is connected in the Connect Call block. For more information about Find Agent and Connect Call, see How to Create a Basic Scenario. As a reminder, define all conditional exits.

How to Create a Voice Scenario Survey

Surveys are useful tools to get feedback from customers about your company, targeted topics, and so forth.

This article explains how to create a voice scenario survey for the purpose of collecting customer satisfaction information. Note that this scenario is slightly different from the Customer Survey template provided in the Contact Center Administrator application.

Scenario Example

Click the following link to download an annotated version of this voice scenario example.
File:App Voice Scenario Survey.zip

For instructions on how to import this file into your contact center, see the *Contact Center Administrator Guide*, section Scenarios Overview > How to Export and Import Scenarios.

For general information about scenarios, refer to section Scenario Builder Overview.

Scenario Flow

The purpose of this scenario is to solicit customer satisfaction feedback from a voice interaction and collect this information for reporting purposes. This is accomplished by using the following scenario blocks:

- Menu
- Set Variable
- Collect Digits
- Save Survey Response

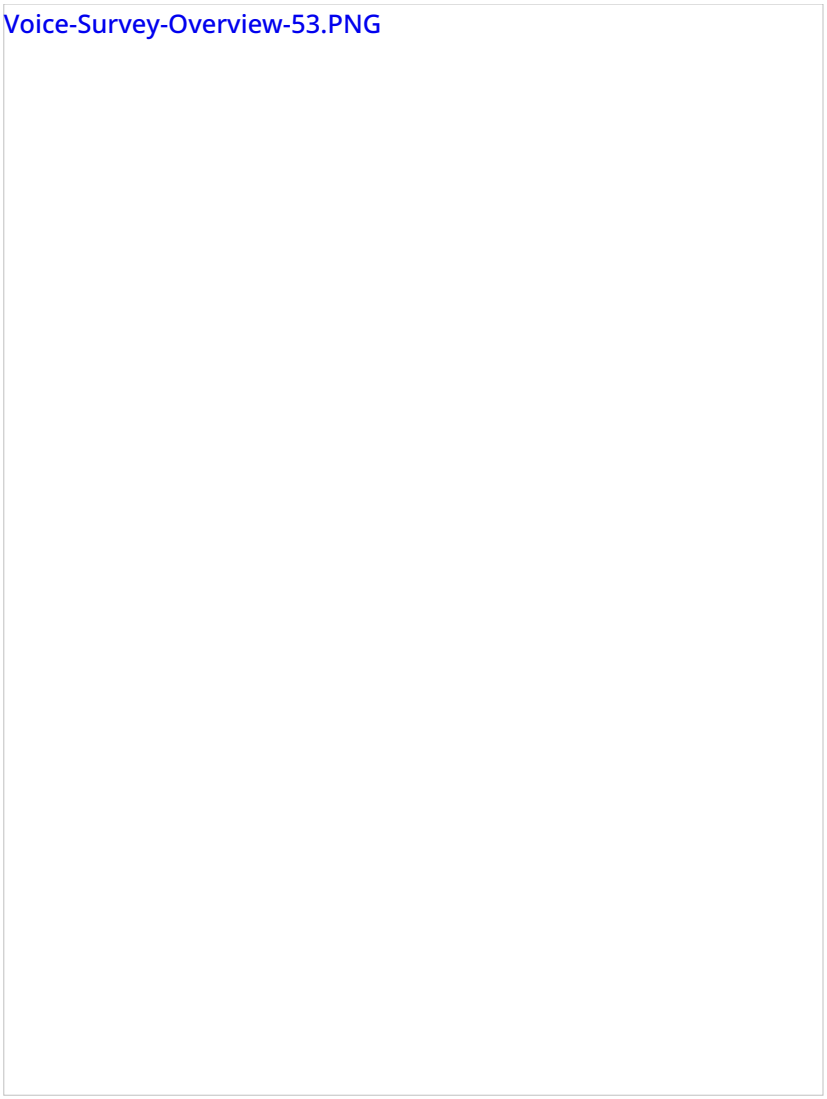
Designer's note: This scenario is triggered when the Start Another Scenario block is invoked from the Target Disconnected conditional exit of a Connect Call block in a different scenario. That is, the agent **must** be the one who ends the call, so it is advisable to have her ask the caller if they would like to participate in a survey and then instruct the caller to stay on the line.

Please note this is an example scenario only and not intended for production use. All conditional exits should be defined with actions for production use.

Scenario Overview

The diagram shown illustrates what the complete scenario looks like when designed in the Scenario Builder application.

Voice-Survey-Overview-53.PNG



Action 1: Tell the Customer They Are Taking a Survey

This scenario begins after interaction with the agent ends, so you will want to inform the customer they have been directed to take a survey. In our example, the survey contains three questions, and we use the Play Prompt block to convey this point.

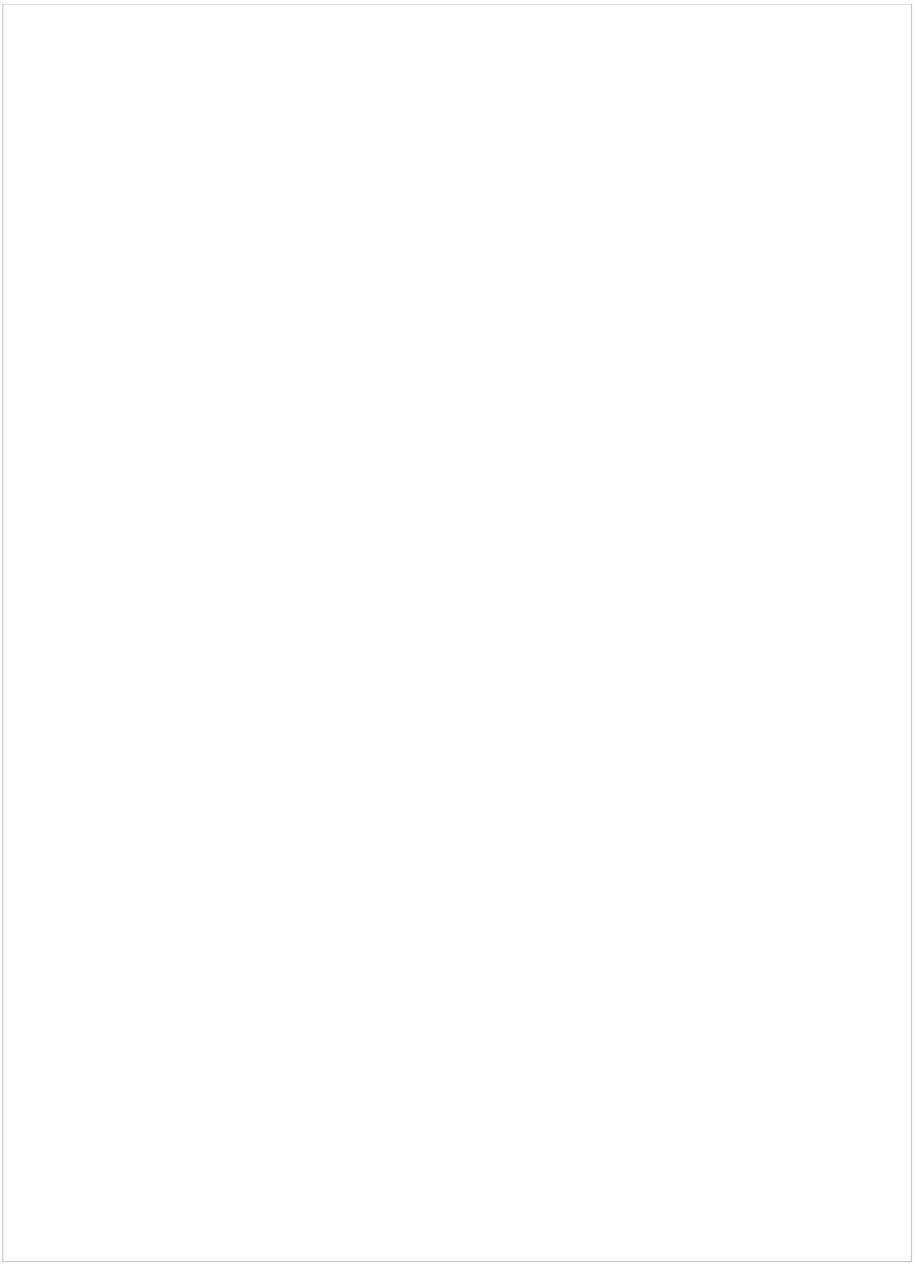
The Play Prompt block's settings

Action 2: Begin Collecting Information with the Menu Block

Next, you will configure a Menu block. In our example, we first configure the Prompt to play, which asks the customer if the problem they experienced was resolved and then instructs them to press a telephone key for their answer.

Next, we configure the telephone key valid options, which are actually conditional exits. In our example, we configure **1** to mean *Resolved* and **2** to be *Not resolved*.

Finally, we configure an Input timeout for 15 seconds, and the number of Retries to three. Doing this allows the scenario to proceed to the Exit block that follows the Menu block in the flow if no input happens within 45 seconds (i.e., the scenario will end).



How to configure your Menu block

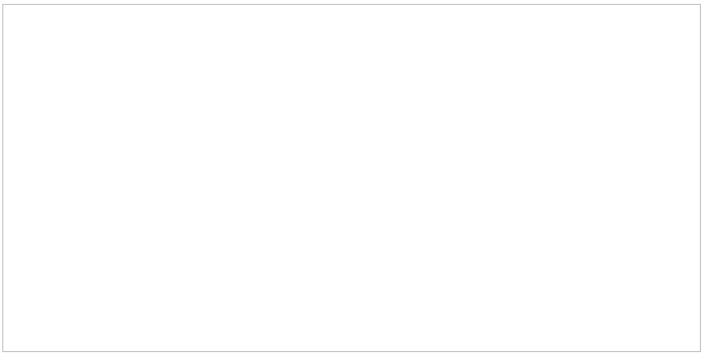
Action 2a: Use the Set Variable Block to Pass Menu Values

If the customer has pressed either of the telephone key valid options, we want two things to happen:

- We want to collect the information about which key was pressed.
- We want the customer to answer the remaining two questions from our three-question survey.

In order to collect key information for each of the valid options, we use the Set Variable block. In both instances, we create a variable named *first_call*. On valid option **1**, we set the value of *first_call* to 1, and on valid option **2**, we set the value to 2. The variable *first_call* ultimately is passed to the Save Survey Response block and will collect the value corresponding to the option the customer choses.

After collecting the key information, we want the scenario to progress to the remaining two questions. This is accomplished by using Goto blocks to jump past the Exit block to the first of the two Collect Digits blocks.



The Menu block's valid choices configured with Set Variable blocks

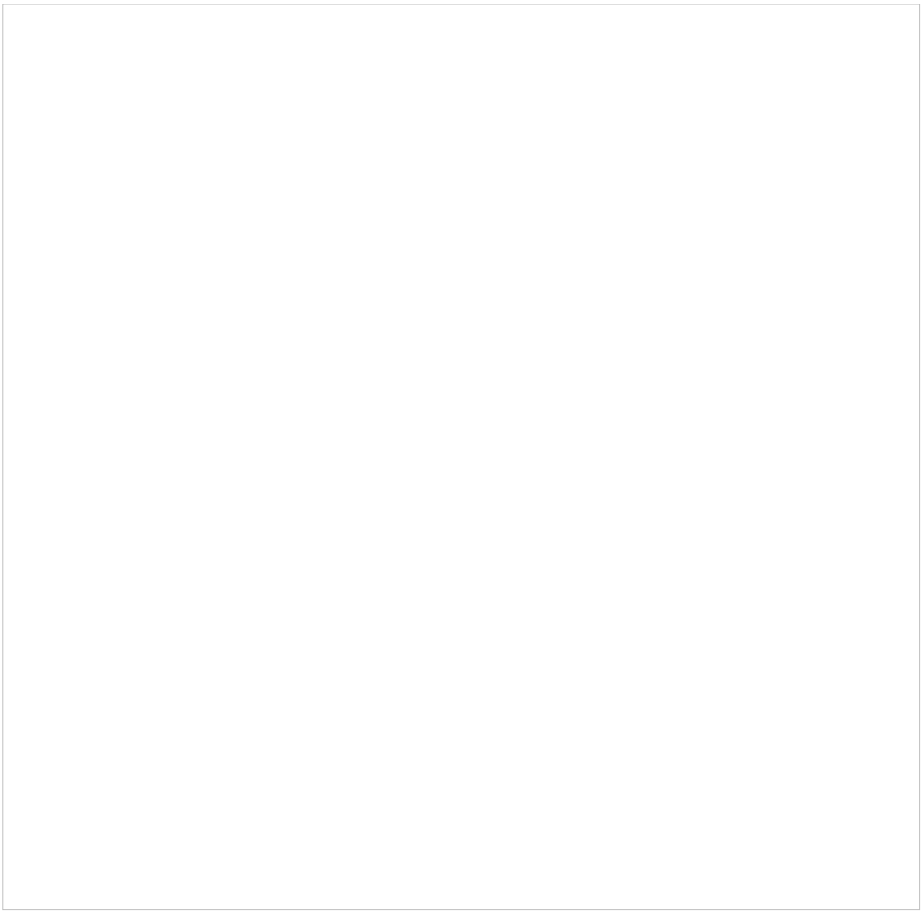
Action 4: Use Collect Digits to Gather More Feedback

You will configure the remaining two questions in two separate Collect Digits blocks; in the scenario flow, these blocks execute consecutively.

Each Collect Digits block plays a prompt that asks a different question. In our example, the first Collect Digits block prompt asks the customer about the helpfulness of the agent and the second asks how likely they are to recommend our service.

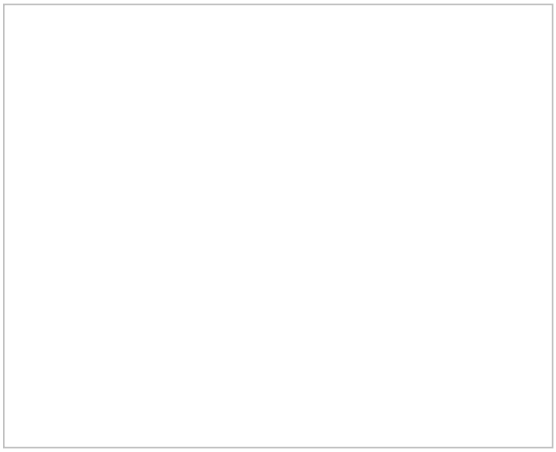
Both blocks instruct the customer to enter numbers ranging from zero to 10 in order to answer the questions, so for each block, we configure the Max number of digits to expect to **2**. Then, for the first block, we configure the Name of the variable to store the result as *contact_satisfaction*; for the second block, we configure the Name of the variable to store the result as *NPS_raw*.

Depending on the numbers the customer enters, both the *contact_satisfaction* variable and the *NPS_raw* variable pass the values to the Save Survey Response block.



One of the two Collect Digits blocks

Although the Collect Digits block does not allow you to configure conditional exits per configured option, like with the Menu block, it does contain regular conditional exits and timeout options that you can adjust to your liking.



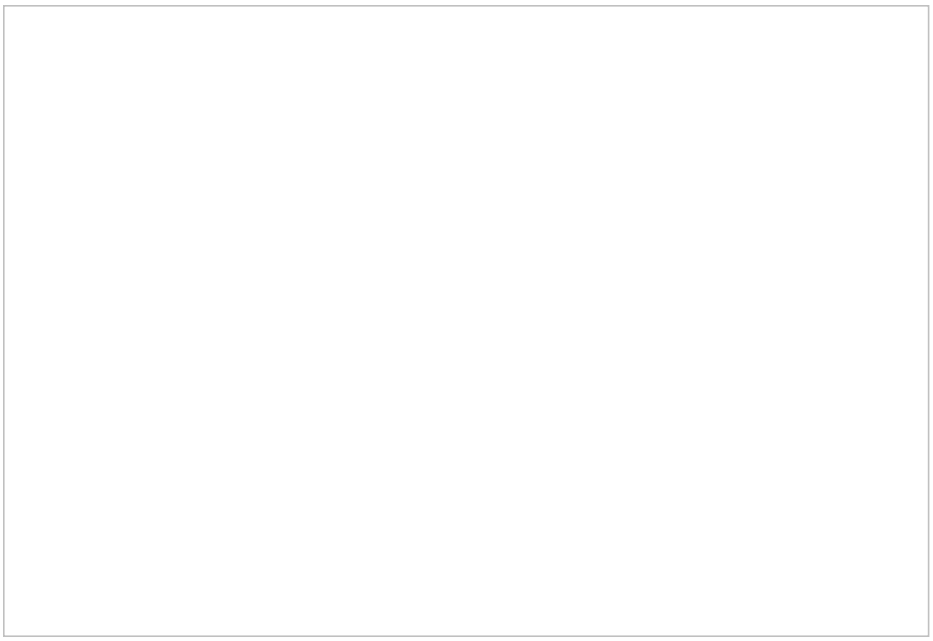
Collect Digits conditional exits configured with Exit blocks

Action 5: Collect All Responses in a Save Survey Response Block

Before ending the scenario, you will want all values collected in the variables you created to be saved; this is accomplished by entering the variable names in a Save Survey Response block. In our example, we enter the variables as follows:

- In the Issue was resolved field, we enter the variable from the Menu block, *first_call*.
- In the Contact satisfaction field, we enter the variable from the first Collect Digits block, *contact_satisfaction*.
- In the Net Promoter Score data field, we enter the variable from the second Collect Digits block, *NPS_raw*.

From here, the numbers the customer entered will be available for reports.



The Save Survey Response block containing all survey variables

Action 6: Play a Prompt to Thank the Customer

Finally, after the customer answers all questions and the answers are recorded in the Save Survey Response block, we use the Play Prompt to thank them for participating in the survey then end the scenario with an Exit block.

Play Prompt then Exit

Suggested Reading

In order to better understand the blocks and variables used in this scenario, we recommend reading the following articles:

- Scenario Builder Basics
- How to Create a Basic Scenario
- How to Create a Voice Scenario That Distributes Surveys to a Percentage of Random Customers

And finally, don't forget to save your work early and often!

How to Use Conversational IVR in a Scenario

To invoke Bright Pattern's conversational IVR feature through a scenario, you simply need to modify an inbound voice scenario to include bot-specific scenario blocks such as Chat Bot Select Account and Ask a Bot, along with the Play-Listen scenario block. These blocks work with integrated Speech-to-Text technologies to recognize and transcribe voice input, as well as with integrated chatbots to analyze voice input, offer suggested replies, and intelligently route callers to agents.

This article will show you how to include these blocks in a basic voice scenario to:

- Greet the customer and listen for a response
- Reply up to three times to the customer using voice prompts
- Collect information from the customer
- Find an agent and connect the call when it's determined that the bot/IVR is not being helpful
- Screen-pop the collected data and/or IVR conversation transcription to the agent upon accepting the interaction

Prerequisites

If you have not already done so, please complete these steps before proceeding:

- Create a Watson Assistant or Amazon Lex bot
- Add a bot/chat suggestions engine integration account
- Set up a Speech-to-Text engine through your service provider
- Add Speech To Text integration account
- Set up a Text-to-Speech engine through your service provider
- Add Text To Speech integration account
- Download and import our bot scenario template: File:App Conversational IVR Example.zip

Scenario Flow

Scenarios are made up of scenario blocks that specify how interactions are processed. Detailed descriptions of Bright Pattern scenario blocks are available in this guide's Scenario Block Definitions section.

The following sequence of actions illustrates how scenario blocks function in a bot-driven voice scenario that invokes conversational IVR. This example scenario uses an integrated Watson Assistant chatbot and Google Speech-to-Text.

Example voice scenario with Play-Listen block in Scenario Builder

Action 1: "Set Prompt Language" specifies which language the voice of your conversational IVR will use

Conversational IVR utilizes voice prompts to speak to callers on the phone. You can set any language for all the voice prompts to be used in this scenario by editing the properties of the Set Prompt Language scenario block.

Set Prompt Language properties

In **Language**, select the desired language for prompts. In our example, we selected "English - United States."

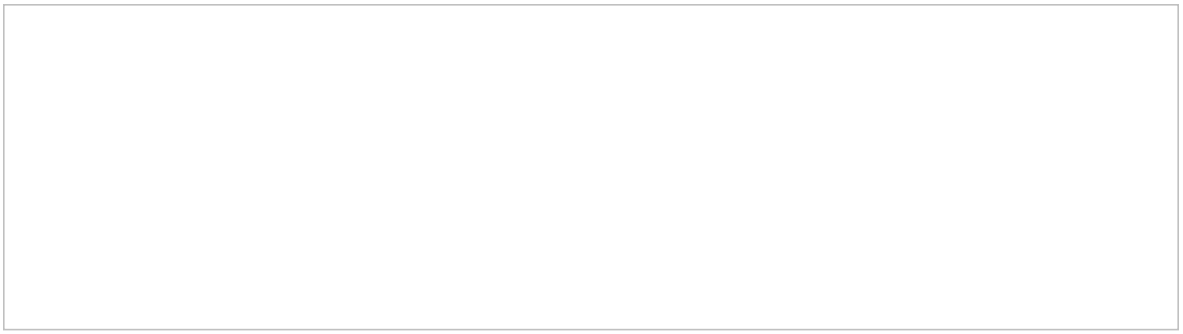
In **TTS voice**, select the Text-to-Speech (TTS) voice type to be used. In our example, we selected "en-US-standard-C."

These settings allow prompt text to be read and spoken over the phone in English.

Action 2: "Set Variable" counts how many times the customer talks to the bot

In this example, we want the caller to talk to the chatbot no more than three times before the caller is routed to an agent. Limiting the interaction in this way prevents the customer from abandoning the call when the bot becomes unhelpful.

With the Set Variable block, we are counting the number of attempts to interact with the bot. The counter begins at zero, so we specify a value of "0."

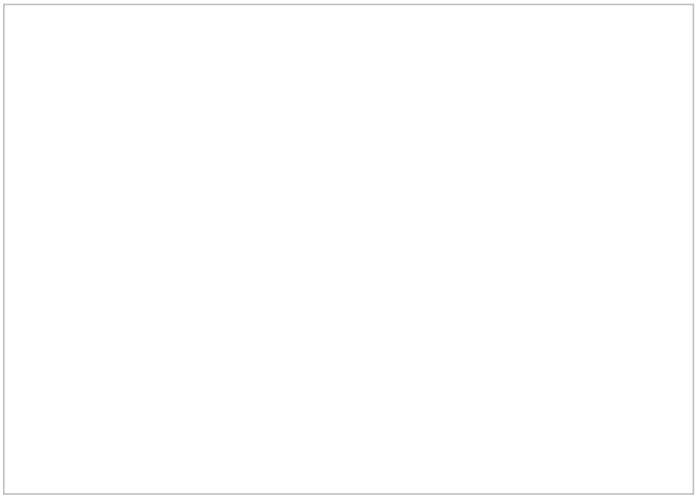


Set Variable properties

Action 3: "Chat Bot Select Account" specifies the integration account for your bot

All scenarios that incorporate bots will include the Chat Bot Select Account block. In this block, you are selecting a bot/chat suggestions engine integration account that has already been configured for your contact center.

It's possible for your contact center to have many integration accounts, so it's important to specify which one will be used for this scenario. If you do not see the desired account listed, go back and add an integration account.



Select an integration account

Action 4: "Play-Listen" talks to the caller and collects a response

The Play-Listen block uses text-to-speech and speech-to-text technologies to convert bot response text into a voice prompt and to recognize and transcribe the customer's reply into a text message.

Specify or create a voice prompt so that the conversational IVR can "talk" to the customer. In this example, the customer will be greeted by the prompt you specify here.

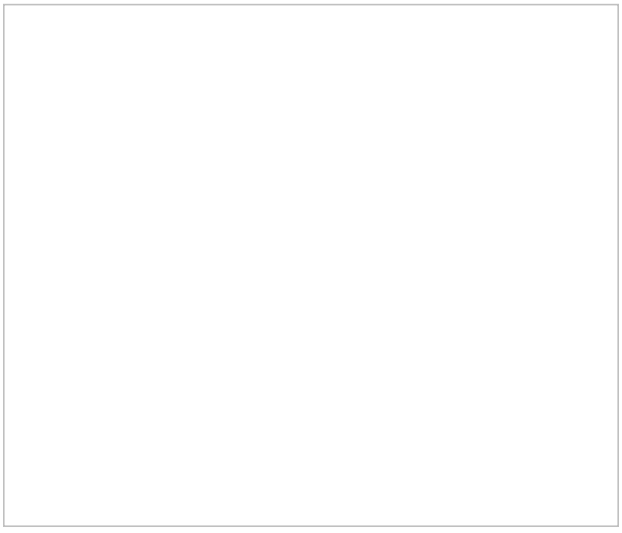
The *transcriber* is the Speech-To-Text integration account that you have already configured for your contact center.

In this example, we set the *recognized phrase* (i.e., the variable) to be "user_phrase," which is the variable that receives what the customer says. We use this variable in later blocks in this scenario to send the customer's response to the agent in a text message (see "Internal Message"). You can add any variable you want, as long as you use the same variable name later.

Confidence is the variable that shows how much of the customer's vocal input the STT engine understands. As with the recognized phrase variable, you can enter anything you want here, as long as you use it consistently later. Your STT engine conveys confidence numerically, where the higher the number (e.g., "0.999") is, the more confident the STT engine is that it has recognized vocals correctly. A low number (e.g., "0.001") shows that the engine has very low confidence in its vocal recognition. For example, if the customer says, "I need a new phone," and the engine hears, "I need a sea foam," the confidence will be likely very low. The confidence level is available in your service engine (e.g., your Watson Assistant).

Max timeout is the number of seconds that the conversational IVR will wait for the caller's response before timing out and moving through the scenario.

In the conditional exits of Play-Listen, you can define what the scenario should do if the caller uses DTMF touch-tones or if the caller is silent.



Use Play-Listen to "talk" to the customer and collect data

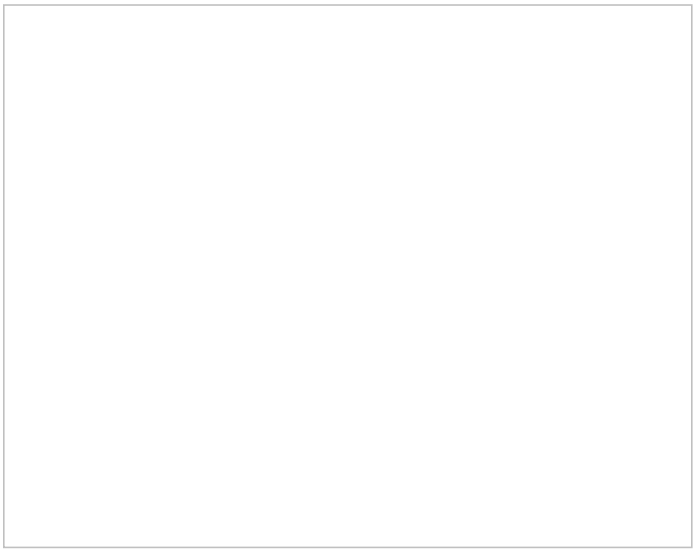
Action 5: "Ask a Bot" lets the bot talk to the customer

The Ask a Bot block allows the bot to automate the conversation before an agent is connected. It delivers the bot's raw response (i.e., suggestions) to the customer.

Notice that in the *Message* field, we've plugged in variable $(user_phrase), which is the same variable we set earlier in the scenario. Placing user_phrase here allows the scenario to pass the caller's vocal input (i.e., his or her spoken phrase) to the integrated chatbot (in this case, Watson Assistant), for processing in the bot's dialog flow. This processing creates *suggestions*, which can be passed to the agent on Agent Desktop.

A *raw response* is configured in your Watson Assistant dialog flow settings; it's unrelated to Bright Pattern. In this example, we've left this field empty.

In *timeout*, you set the number of seconds to wait for the bot's suggestions before timing out.



Get suggestions from the bot

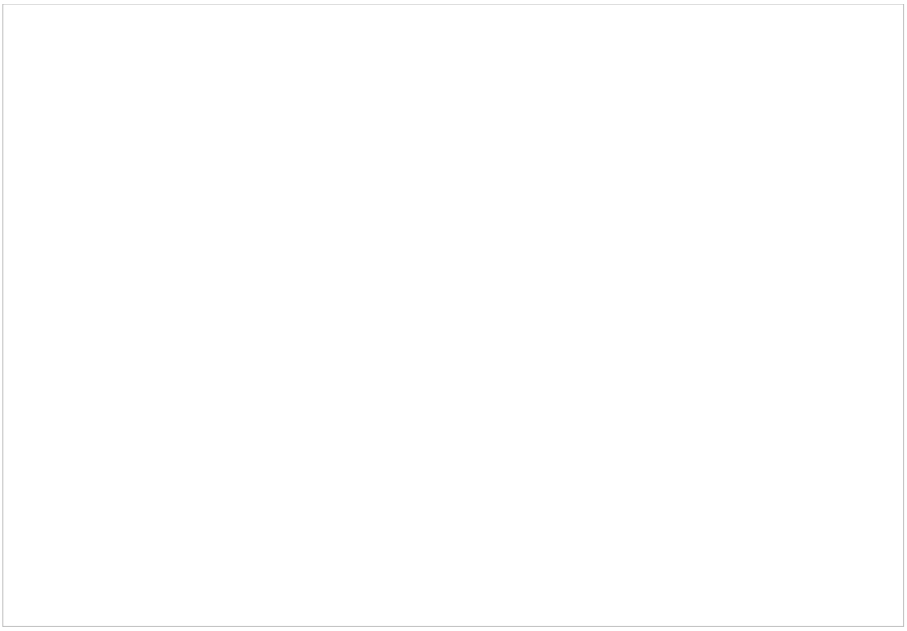
In the Ask a Bot block's conditional exits, we have dragged over some other blocks that tell the scenario what to do if this block fails, times out, or if there's no data. See the scenario template for more detailed comments on what we did in this example.

Action 6: "If" defines what to do if no data is collected by the bot

Within Ask a Bot, there is a conditional exit called *No Data*. Here, we dragged over some other scenario blocks to define what the scenario should do if the bot fails to collect information from the caller.

First, use *Set Variable* to set variable name "Attempts" to *=$(Attempts)+1*. This is used for counting how many times the bot attempts to talk to the caller and offer a suggested reply.

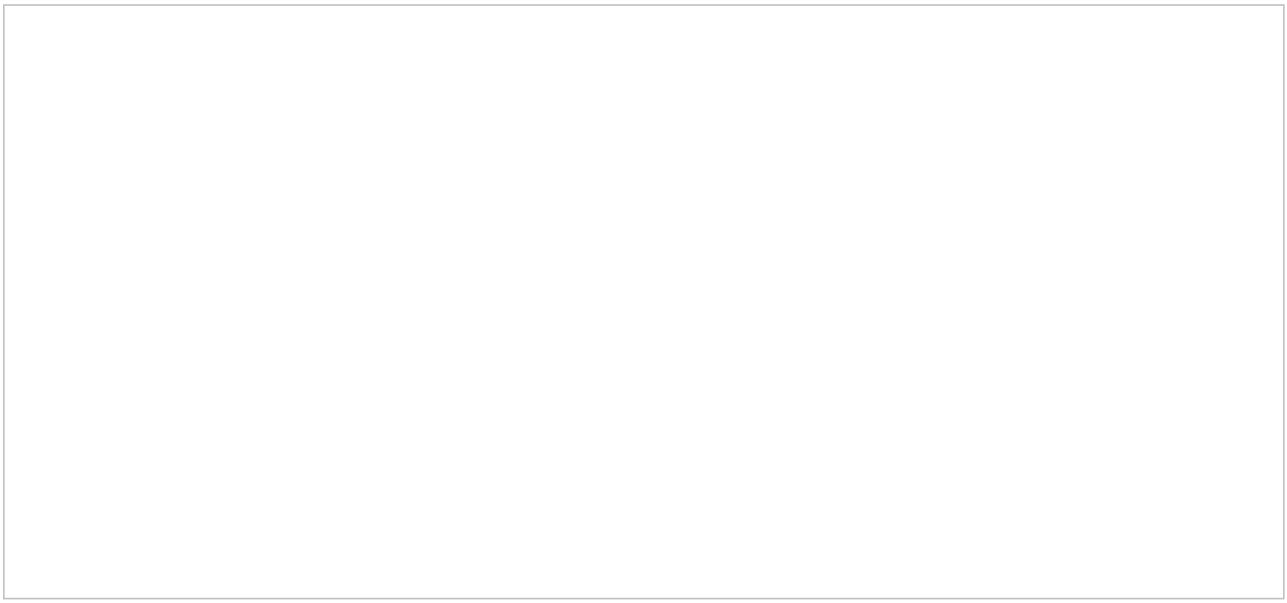
Then use If to add a condition that says if the bot tries to talk less than three times, try again. Remember, in Action 2 of this procedure, we used Set Variable to start counting bot step interactions.



Define what to do if the bot attempts to talk to the caller less than three times

Under the *If* block, we added another Play-Listen block that will be triggered if the condition is met. In this Play-Listen block, we define that if the bot has to make another attempt to collect data from the caller, then the prompt is, "I'm sorry, I didn't understand you...."

If that still fails to collect information, the scenario uses Goto to find an agent and connect the call. The Play Prompt block tells the caller that the bot can't understand, and thus, an agent will be connected.

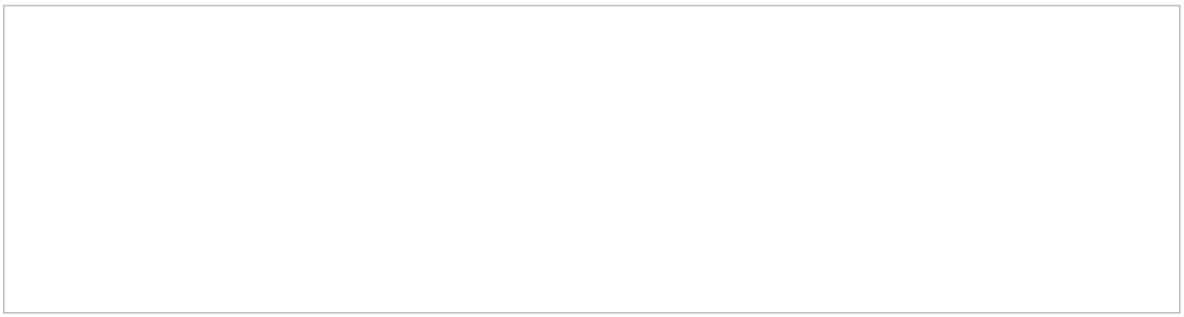


The condition has been exceeded and a prompt is played before an agent is connected

Action 7: "Set Variable" defines the bot's suggestions

Set Variable is used for defining the variable name of the bot's responses (i.e., suggestions). Suggestions align with what you have configured your Watson Assistant, or other chatbot, to say.

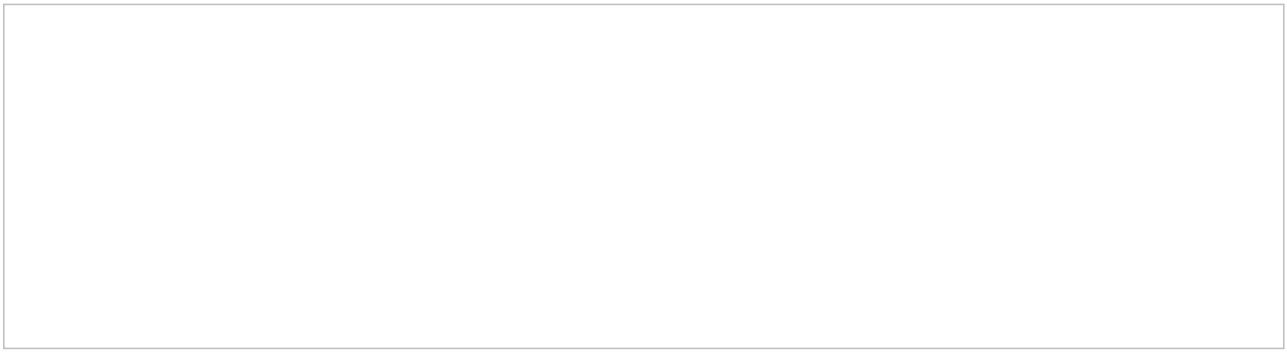
In this example, we named the variable "bot_phrase" and gave it a value of "$(suggestions[0].msg)." Later, when we pass the conversational IVR transcript to the agent via screen-pop, we will be invoking the conversation via "bot_phrase" and "user_phrase" variables.



Set Variable properties

Action 8: Another "If" will find an agent if the caller speaks

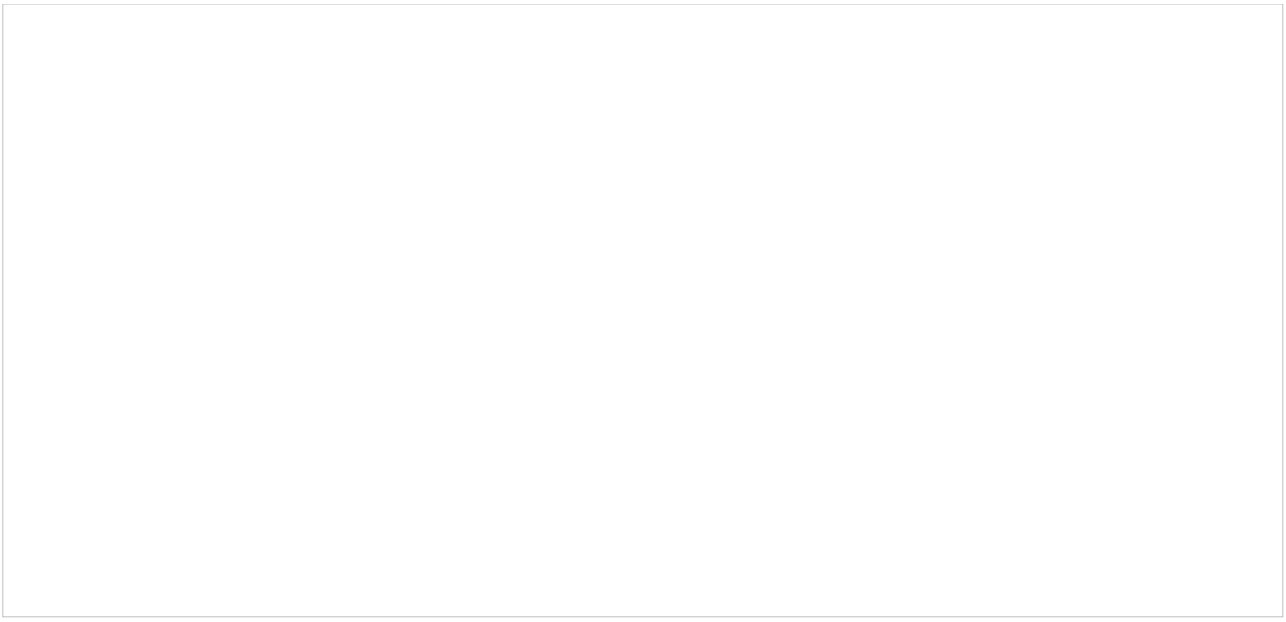
In this example, we added the condition that scenario will branch if the "user_phrase is not empty" (i.e., the caller is talking).



Ask a Bot branches and finds an agent in response to what the caller said

Action 9: "Web Screen pop" passes the conversation transcript to the agent

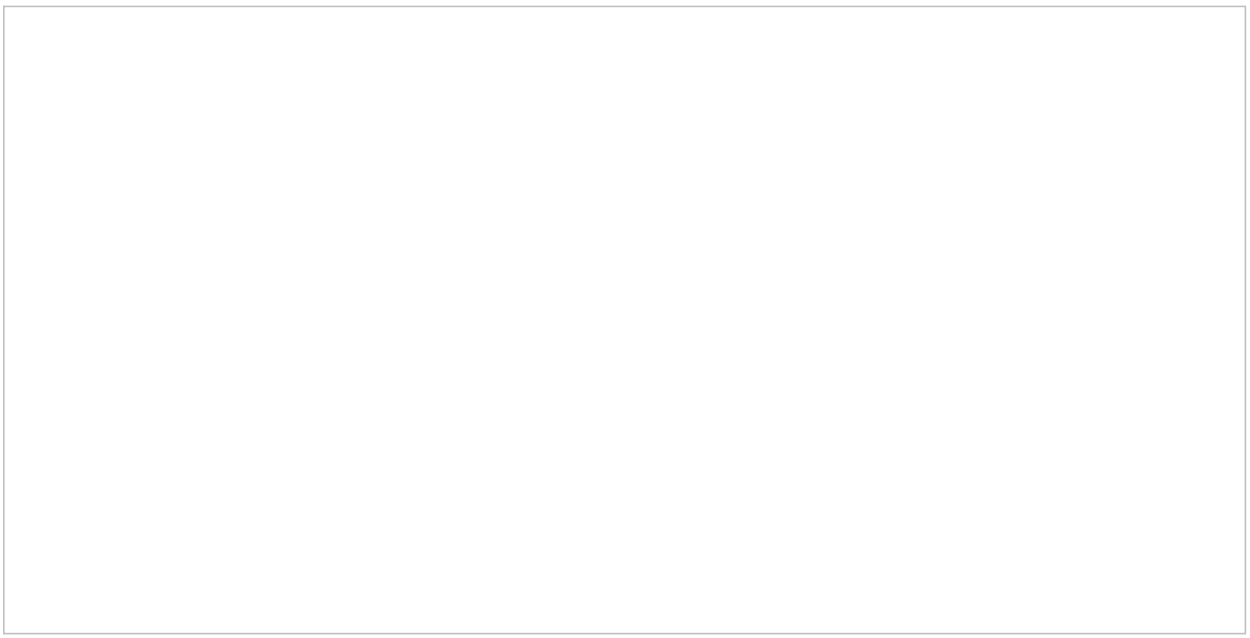
In the Web Screen pop properties, we defined what text to display to the agent upon accepting the call. In the "Text to display" field, we entered the variables "user_phrase" and "bot_phrase" in basic HTML. This will appear as plain text to the agent on Agent Desktop.



You can screen-pop the bot and customer's conversation transcript to the agent

Action 10: "Find Agent" looks for a skilled agent

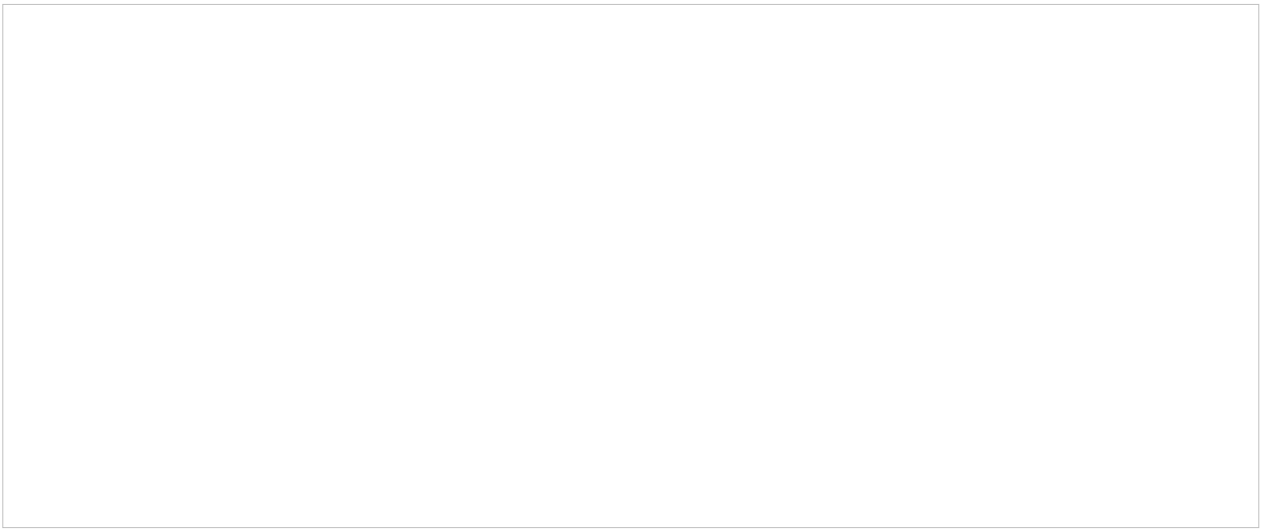
The Find Agent block looks for the next available skilled agent to accept the call. You can use Find Agent to set wait times and send the customer messages about estimated waiting time (EWT).



The bot couldn't help the customer is fewer than three steps, so the scenario looks for an agent

Action 11: "Connect Call" always comes after "Find Agent"

Although Find Agent uses criteria to identify the proper agent to receive the call, it does not have a "delivery" mechanism. Connect Call is the delivery mechanism for the interaction.



You can either specify a destination for the call or let the system find the next available agent

Action 12: "Exit" ends the scenario

The Exit block completes the scenario. Without it, the scenario will loop through this configured flow until the customer ends the call.

Last Action: Save!

Be sure to click **Save** to apply your changes. Note that closing your web browser window or tab will close Scenario Builder without saving, and you will lose your work.

Recommended Reading

For more information on inbound voice configuration and bot setup, see these Bright Pattern tutorials:

- Scenario Builder Basics
- AI and Bot Tutorials
- How to Create a Watson Assistant
- How to Create an Amazon Lex Bot
- Inbound Voice Service Configuration

Scenario Example

This section provides a simplified example of a typical scenario for processing inbound service calls.

Imagine contact center operations of a security equipment company called All Safe. The contact center provides two services: one for product sales and the other for support. Both services are provided within typical business hours and are available in two languages, English and Spanish. The company's contact center has one general service number and uses interactive voice response (IVR) technology for language and service selection before distributing calls to qualified agents.

To support this operation, the following resources are configured in the system:

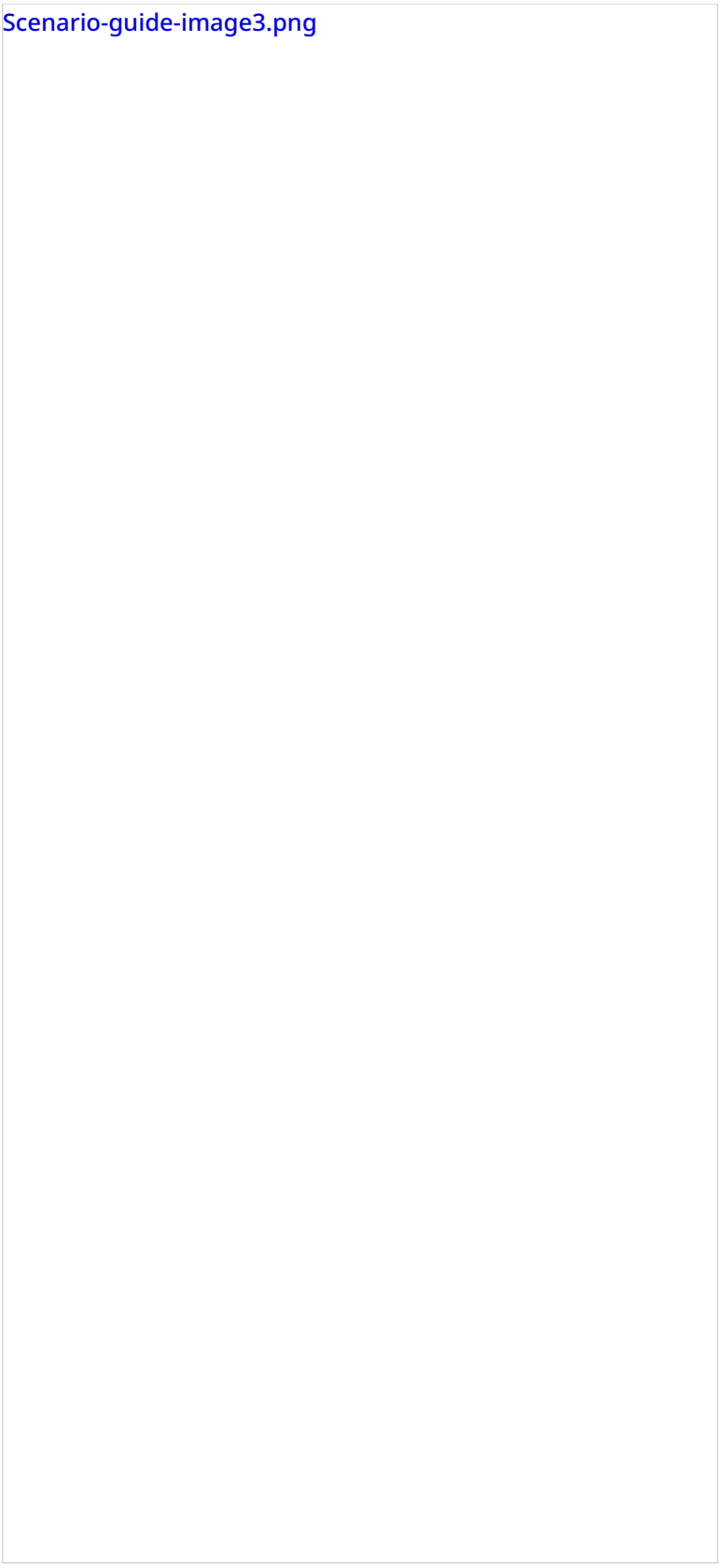
- Services: **Sales** for sales calls, **Support** for support calls, and **General** for unqualified calls and general inquiries
- An auxiliary skill group: **Languages** with skills **English** and **Spanish**
- A team of agents specializing in product sales and associated with the services **Sales** and **General**. All agents have the corresponding default service skills, all have the **English** skill, and some also have the **Spanish** skill.
- A team of agents specializing in product support and associated with services **Support** and **General**. All agents have the corresponding default service skills, all have the **English** skill, and some also have the **Spanish** skill.
- Hours of operation (HOP) for all services.
- An access number that customers will use to call the services.

A simplified scenario for processing calls made to All Safe may look like this:

1. First, the call arrival time is checked against the contact center operational schedule. If the call arrives outside of the business hours, an announcement is played back, prompting the caller to call again during business hours. The call is then disconnected.
2. If the call arrives within business hours, a general greeting is played first.
3. The caller is prompted to select a language.
4. Based on the caller's input, the corresponding language skill is set as one of the agent selection criteria, and the language of the subsequent voice prompts is set to match the caller's preference. If the caller does not provide any input within a timeout, the English language is set for both the language skill and voice prompts.
5. The caller is prompted to select the service.
6. Based on the caller's input, the corresponding service skill is set as one of the agent selection criteria. If the caller does not provide any input within a timeout, the **General** service skill is selected.
7. The call is then queued while the system looks for an available agent who has both selected skills.
8. As soon as such an agent becomes available, the system delivers the call to the agent extension. (If other calls are waiting for the same skill set, they are distributed to agents in the order of their arrival.)
9. If the selected agent does not answer, the call is returned to the service queue to wait for the next available agent with the matching skills.

When translated into the Bright Pattern scenario language, the scenario description appears in the Scenario Builder application as shown.

Scenario-guide-image3.png



Once the scenario is defined, a dial-in scenario entry will have to be created in the configuration in order to associate this scenario with the corresponding external access number.

< Previous

All Scenario Templates

There are many ways to build a scenario, but sometimes it is easier to start with a template and modify it later. The following custom templates, which include instructional comments and explanations, are provided for you to download and import into your contact center.

Templates are provided as .zip files. Click to download.

Chat

Have a chatbot respond to customer two times before transferring to agent

File:App GeneralChatBotScenario.zip

Pop a saved contact's existing cases to the agent

File:App Screen Pop Customer Cases.zip

Sending Automatic Email Replies to Customers Who Use the "Leave a Message" Chat Form

File:App After-Hours Send an Email Reply to a Chat Customer.zip

Configure a chat scenario to use a Microsoft Azure Web App bot

File:App Azure Web App Bot Example.zip

Voice

Block incoming calls from specific phone numbers

File:App How to Block Incoming Calls from Specific Numbers.zip

Create a voice scenario survey

File:App Voice Scenario Survey.zip

Distribute surveys to a percentage of random customers

File:App How to Distribute Surveys to a Percentage of Random Customers.zip

Identify customer using Microsoft Dynamics 365 data

File:App MS Dynamics Search and Pop.zip

Redirect Calls Economically with a Single-Step External Transfer Option

File:App Menu with a Single-Step External Transfer Option.zip

Route caller to the last agent with the option to leave a voicemail

File:App FindLastAgentVM (2).zip

Skill-Based Call Routing with an Auto Attendant Choice

File:App Skill-Based Call Routing + Auto Attendant.zip

Use conversational IVR in a voice scenario

File:App Conversational IVR Example.zip