Genesys Designer Help

Designer current

12/29/2021

Table of Contents

Genesys Designer Help

> **Important**
>
> - **This help content is specific to the 8.5 release of Designer.**
> - If you are using the latest version of Designer (version 9.0), the help topics for that release are found in the Designer User's Guide in the Genesys Engage Multicloud Resource Center (you can check your version of Designer in the top-right corner of the Designer interface).
> - Contact your Genesys representative if you have questions about features that are available in your version of the software.

Welcome to the Genesys Designer Help. Genesys Designer is a web-based tool for developing Self Service (Interactive Voice Response, or IVR) applications and Assisted Service (Routing) applications that run on the Genesys platform.

What's in this Guide?

This guide introduces you to the Genesys Designer user interface and explains concepts and procedures to help you use this software in your contact center.

If you are new to Designer, or you just need a quick refresher on key concepts, refer to the Genesys Designer Quick Start Guide.

Once you have mastered the concepts in the Quick Start Guide, take a look through this Help to learn about the following topics:

- Getting Started - This chapter introduces you to Designer and explains the user interface, security permissions, application phases, and how to save and publish your applications.
- Using the Navigation Bar - This chapter explains the menus that you can access in the Navigation Bar.
- Using Blocks - This chapter provides reference information for the blocks that you can use with your applications.

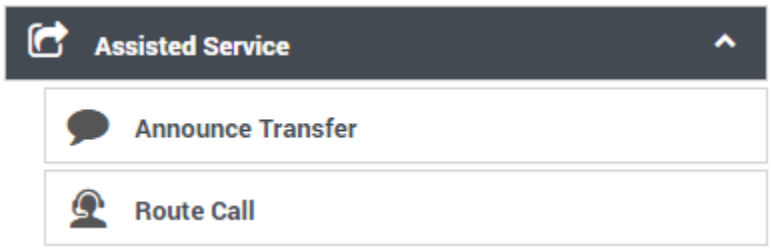
Getting Started

> **Tip**
> Feeling lost? Refer to the Genesys Designer Quick Start Guide for handy information on application structure, application phases, tips and tricks, and much more.

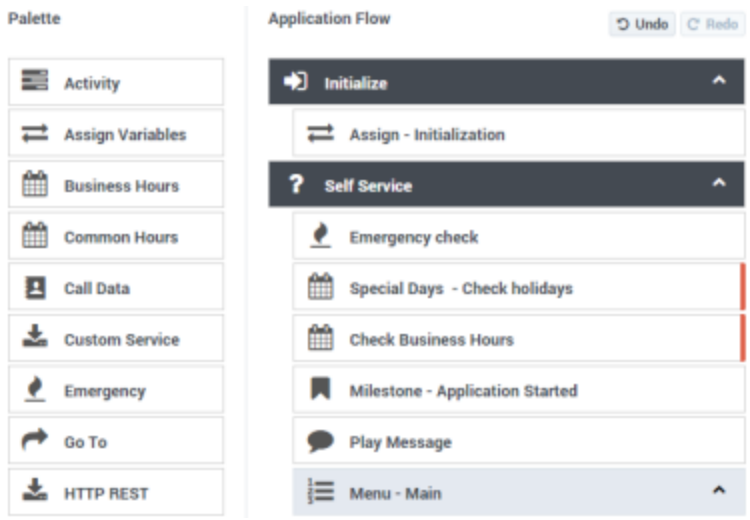
Genesys Designer is a web-based tool for developing Self Service (Interactive Voice Response, or IVR) applications and Assisted Service (Routing) applications that run on the Genesys platform. Each applications consists of a series of blocks that outline the flow and logic of how the application executes.

> **Important**
> Self Service applications are based solely on VoiceXML, whereas Assisted Service applications combine VoiceXML and Routing (SCXML) elements. Genesys Designer provides a seamless development environment that uses both elements, which means users do not need to be aware of these technical details to start developing applications.

To create applications, Designer provides easy-to-use, highly functional blocks that enable common tasks in a simple and straightforward manner. The graphic below shows examples of blocks that you might use in the **Assisted Service** phase, in which an agent helps a customer. The first block controls the announcement to the caller that he is being transferred, and the second block controls the routing function to an agent.



You can view the list of available blocks in the **Palette**. When you are ready to build your application, you can drag blocks from the **Palette** and drop them into the **Application Flow**, which represents the application structure and flow.



Features

Genesys Designer includes the following features:

- Skills-based routing with skill relaxation over time. If an agent with the required skill level cannot be found, you can choose to reduce the required skill level incrementally over time.
- Skill-expression routing to support complex routing parameters.
- Routing to agent groups.
- Priority handling by customer segment and option to increase priority based on response time.
- User data management.
- Customer segmentation based on expressions.
- Cascaded Routing with busy treatments including EWT.
- Invoking external RESTful webservices.
- Audio resource management with support for multiple languages.
- Convenience blocks for Emergency, Business Hours, and Special Days.
- Flow control blocks such as Menu and Segmentation.
- DTMF input collection.
- TTS prompts and intelligent audio stitching.

User Interface

Supported browsers

Designer supports the latest versions of the following web browsers:

- Mozilla Firefox
- Google Chrome
- Microsoft Edge
- Apple Safari

> **Important**
> Microsoft Internet Explorer (all versions) is not recommended for use with Designer. Using non-recommended browsers with Designer can produce unexpected results.

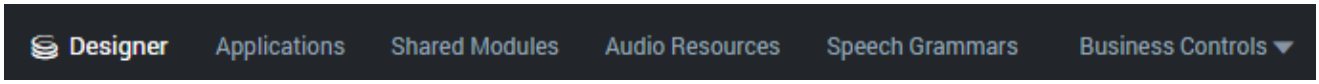
User interface overview

Watch this video to see an overview of the Designer user interface:

Link to video

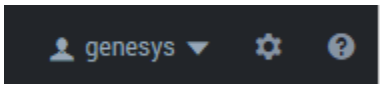
The various elements within the interface are described below.

Navigation bar



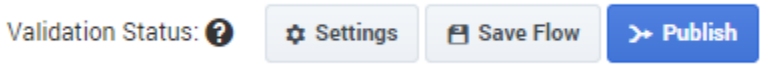
Provides one-click access to Applications, Shared Modules, Audio Resources, Speech Grammars, and Business Controls objects.

Workspace toolbar



Provides buttons for common actions. Click your user name to log off. Click the settings icon to view or modify the global **Caching** settings for certain resources and to toggle certain **Features**. Click the Help icon to access the Designer Help.

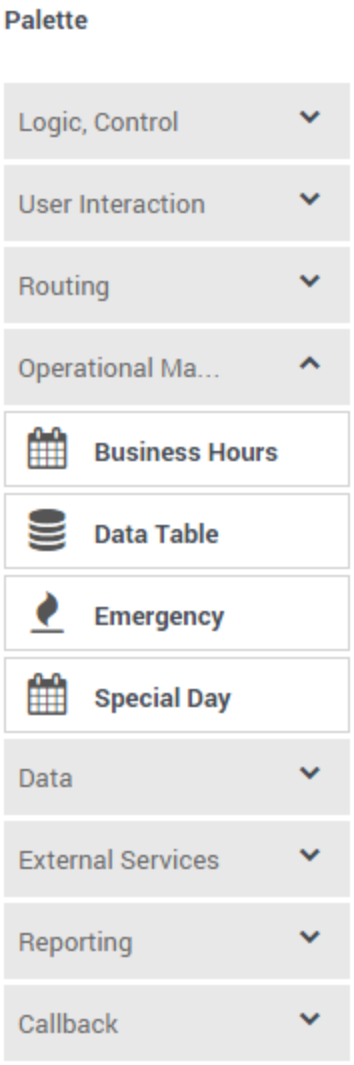
Application toolbar



Provides buttons for common actions. Click Settings to set global settings for your application. Click Save Flow to save and validate your application, or click **Publish** to save and validate your application and prepare it for use by routing engines.

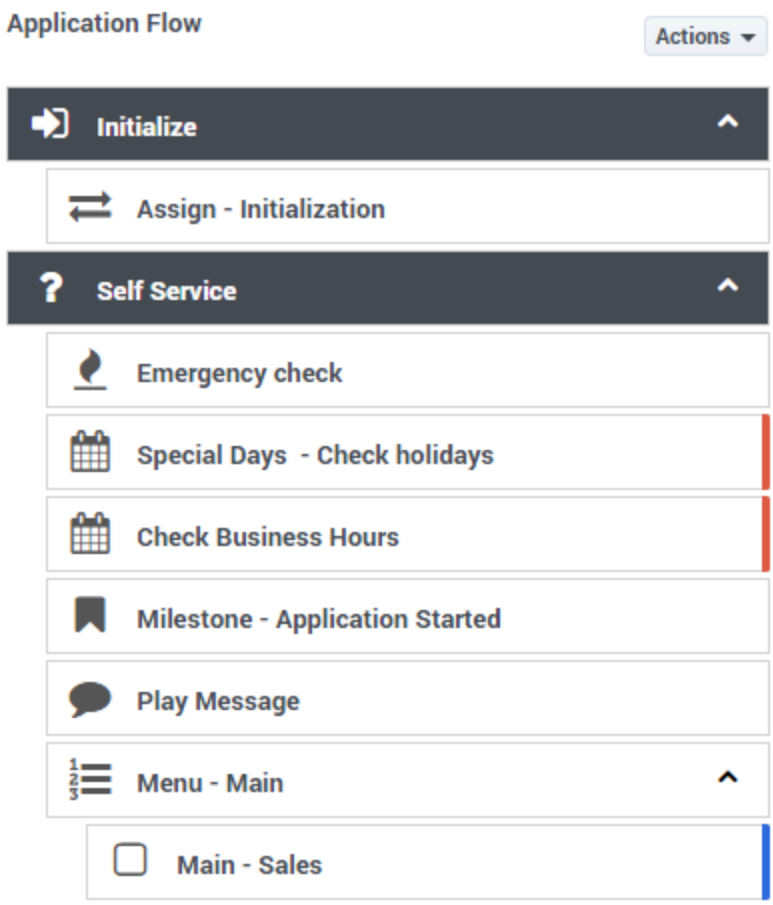
Palette

Provides all available blocks that you can use in your application, sorted by functional grouping:



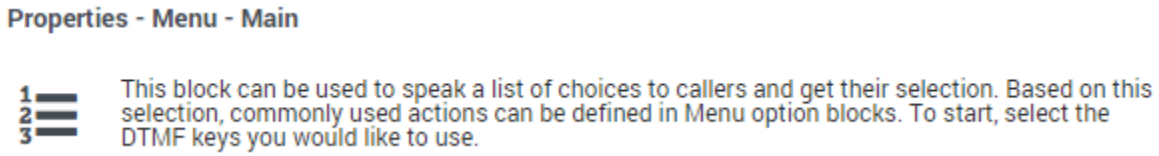
Application flow

Provides the main area to build your application by adding blocks vertically. (See Build Logic for more information.)



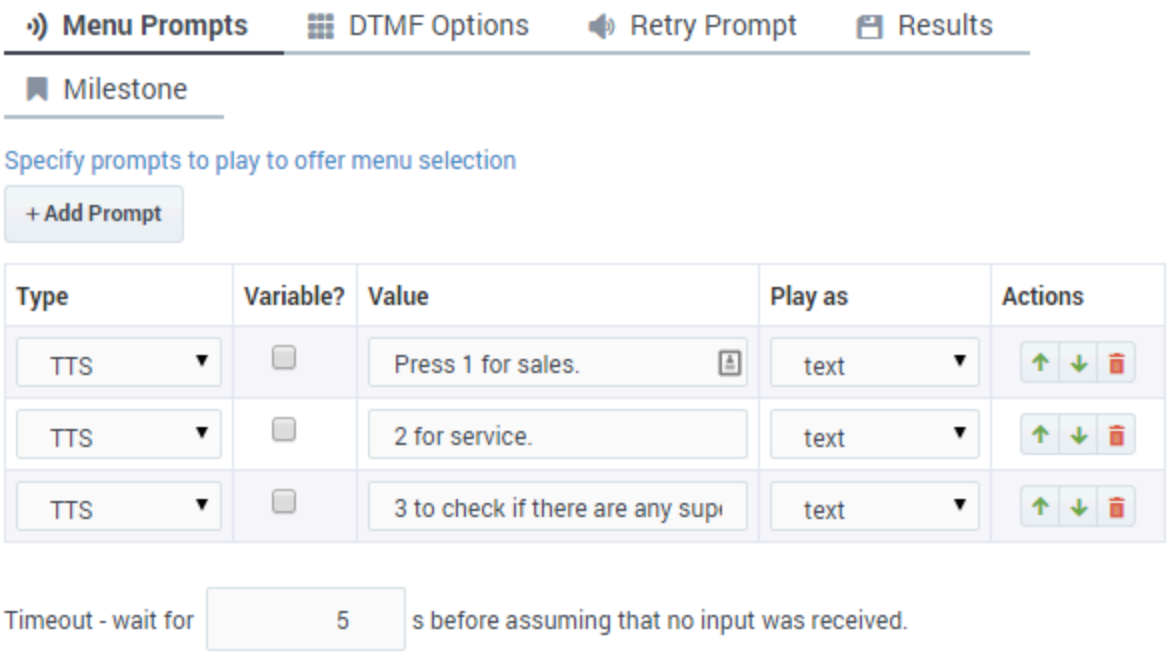
Help pane

Displays help information for the selected block:



Block properties

Displays all properties exposed by a block and provides assistance to set them:



Quick filters

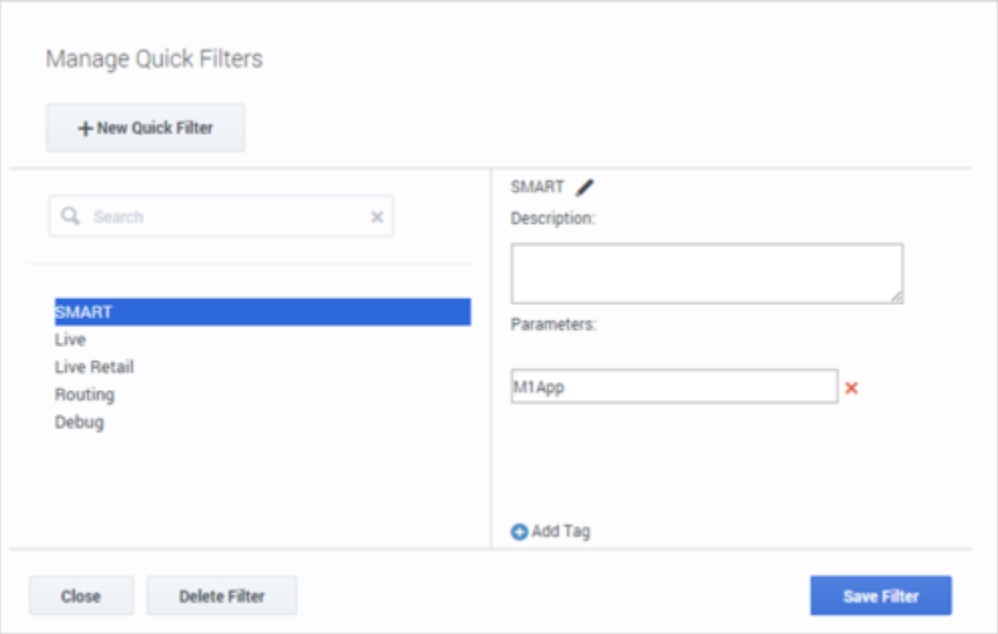
This toolbar enables you to filter a list of resource items by selecting one or more filters that are associated with tags. The list then refreshes to show only those items that match the selected filters.

In this example, the **SMART** and **Debug** filters are selected so that only data tables with those tags are shown:

Note that the filters will display any item in the list that has the associated tag, even if there are other tags associated with that item. If you navigate to a new resource page (such as going from **Data Tables** to **Business Hours**), any selected filters are automatically applied to the new page.

To add, modify, or delete quick filters, click the **Settings** icon to open the **Manage Quick Filters** window. To associate a filter with a specific tag, select it, and add the tag(s) under the **Parameters** section.

In the above example, the **SMART** filter was associated with the "M1App" tag, as follows:



The **Quick Filters** toolbar appears on the following resource pages: Special Days, Business Hours,

Data Tables, Applications, Shared Modules, Emergency Flags, Audio Collections, and Message Resources. The same filters appear on each page, and any filters that you create are visible to other Designer users.

> **Tip**
>
> Tags are a useful tool for keeping resources organized. For consistency, Genesys recommends that you define and use a similar set of tags across your various resource types.

Permissions and Access

Genesys Designer provides layered access roles to ensure that your users only have access that is appropriate for your business needs—such as the ability to make changes to prompts, business hours, or set an emergency routing flag—without exposing control to the overall application logic.

Designer only permits one user at a time to open an application, shared module, or data table for editing. If you try to open one of these resources and it is already in use, Designer shows you which user has the resource locked. You can then choose to open it as read-only or go back to the previous screen.

If a user has opened an application or module for editing and then goes inactive, their browser eventually displays a timeout warning. If they do not respond, the resource is closed and the lock released.

Designer roles

Designer supports the following user roles:

Designer Developer

Designer Developers have the ability to view and modify all resource types within Designer, with the exception of features that are restricted to Designer Administrators (such as managing partitions and user accounts).

Designer Business User

Designer Business Users have limited access to Applications, Speech Grammars, and Shared Modules. They can view these resources, but cannot make any changes to them.

They have full access to Media Resources, Business Controls (which lets them work with business hours, special days, emergency flags, and data tables), and Analytics.

Business Users can also assign phone numbers to applications.

Designer Administrator

Designer Administrators have full access to the Admin settings for Designer, which includes the ability to manage partitions and control which users have access to them. They can view most of the other resource types in Designer, but cannot make any changes to them.

> **Important**
> For full access to all resource types in Designer, a user would need to be assigned to *both* the Designer Administrator and Developer roles.

Designer Analytics

All Designer roles have access to Analytics. Users assigned *only* to the Designer Analytics role can view and modify the Analytics dashboards, but do not have access to any other Designer resources.

Designer permissions

The following tables provide a high-level overview of what each role can do with various Designer resources:

Media Resources

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓	✓		
Create	✓	✓		
Delete	✓			

Message Resources

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓			
Create	✓			
Delete	✓			

Applications

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓			
Create	✓			

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
Delete	✓			
Switch to last snapshot	✓		✓	
Assign phone numbers	✓	✓	✓	
Change block properties	✓			

Shared Modules

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓			
Create	✓			
Delete	✓			

Speech Grammars

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓			
Create	✓			
Delete	✓			

Business Controls

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓	✓		
Create	✓	✓		
Delete	✓	✓		

Blocks

Task	Designer Developer	Designer Business User	Designer Administrator	Designer Analytics
View	✓	✓	✓	
Modify	✓	✓		
Create	✓	✓		
Delete	✓	✓		

Restricted mode during upgrades

During upgrades, Designer continues to provide full service, but goes into a restricted mode that only allows selective modifications to be made. While Designer is in this mode, certain objects (such as applications, shared modules, and workspace settings) are locked for editing until the upgrade is complete. You can, however, continue to make changes to objects such as business controls, audio resources, and grammars.

Application Phases

Applications consist of several common blocks, known as application phases, that divide your application as follows:

- **Initialize** – This phase initializes application-level user variables and parameters to use when the application executes.
- **Self Service** – This phase hosts blocks that provide automated interaction with the caller via speech and/or DTMF.
- **Assisted Service** – This phase hosts blocks that route the call to a live agent, if necessary.
- **Finalize** – This phase provides post-processing and call termination after the call has been serviced.

Initialize

The application initializes during this phase. By default, the following actions execute:

1. Initialize and set up user variables.
2. Load application run-time parameters from external sources.
3. Process call properties (for example, ANI and DNIS) and application run-time parameters. System variables or properties may be initialized internally.
4. If configured, additional processing that was set up by the user.

Self Service

The **Self Service** phase is the IVR portion of the call, which is executed as a VoiceXML application on GVP. This phase attempts to provide automated service and contain the call within an IVR, so there is no need to route the call to an agent. If routing is necessary, this phase collects necessary data from the user through various questions and menus, and then determines how to route the call in the next

phase - **Assisted Service**.

> **Tip**
> To enable call recording on the **Self Service** phase, set the **EnableSSRecording** variable to **true** in the System Variables section.

The following are typical actions that are executed during the **Self Service** phase:

1. Play Messages. These may be pre-recorded audio files or dynamic text spoken using TTS.
2. Check business hours and customize logic based on the outcome.
3. Collect user input using DTMF and ASR.
4. Present choices to callers using Menus.
5. Segment and branch call logic.
6. Call external RESTful APIs and fetch data into user variables.
7. Update user variables and write ECMAScript expressions.
8. Set up and process global commands and hot words.

The **Self Service** phase updates user variables with collected or calculated data. This data is later used by other blocks in the **Self Service** or **Assisted Service** phase.

Call processing might complete during the **Self Service** phase. In this scenario, the application control skips the **Assisted Service** phase and proceeds to the **Finalize** phase. For example, if the business hours check determines that the contact center is closed, the corresponding announcement is played to the caller and the call is terminated.

Assisted Service

> **Important**
> This phase does not appear if you are using an **IVR** type application. See the Applications page for more information on application types.

During the **Assisted Service** phase, the application attempts to route calls to agents. Routing is performed based on data collected in previous phases. For example, target skills are taken from user variables.

The following are typical actions for this phase:

1. Attempt to route the call while playing music or prompts.

2. Call external RESTful APIs.
3. Update user variables.

There may be multiple **Route Call** blocks in sequence. Each **Route Call** block might try to route the call to different targets with different timeouts. For example, it might expand a target by geographical location.

Each **Route Call** block has a timeout, after which the next **Route Call** block in sequence is executed. If any of the blocks successfully routes a call, the **Assisted Service** phase is complete and processing continues to the **Finalize** phase.

Finalize

When call processing is finished, the application goes to the **Finalize** phase to perform post-processing for various scenarios that are based on how the call was completed.

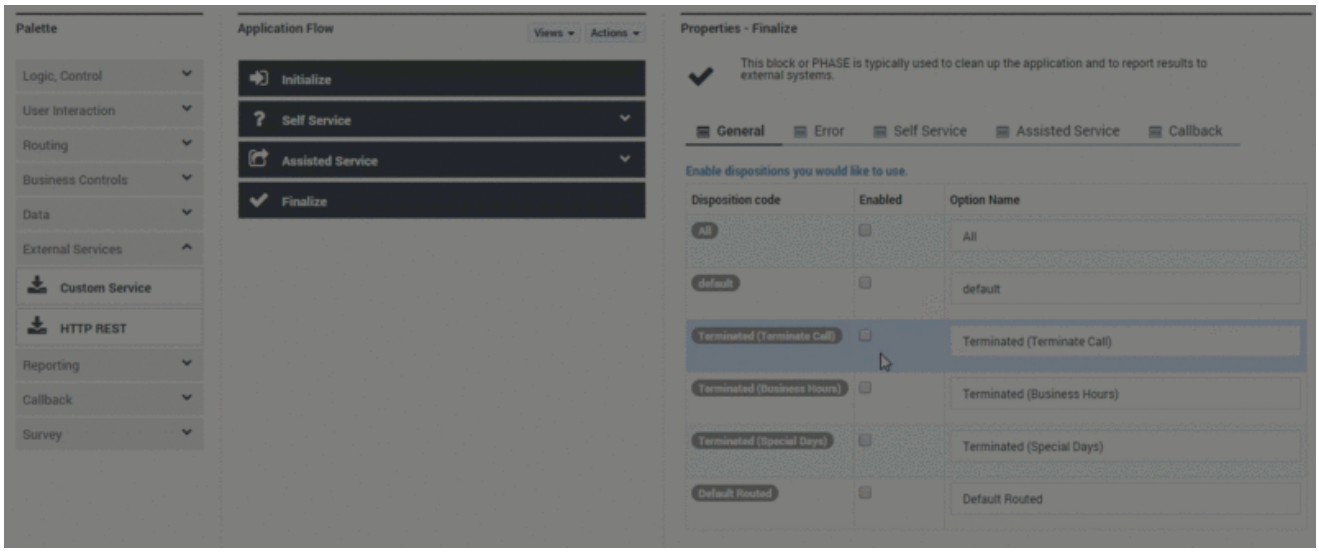
The following are examples of typical scenarios:

1. Call was abandoned by the caller (while in either the **Self Service** or **Assisted Service** phase).
2. Call was completed in **Self Service** phase.
3. Call was routed to an agent in the **Assisted Service** phase.
4. Call was delivered to voicemail in the **Assisted Service** phase.
5. User opted to leave a queue and schedule a callback.

You can also use the **Finalize** phase to submit application data to an external system for reporting metrics, or to select a call disposition code for post-processing.

When you click on the **Finalize** block in the application flow, each of the tabs (**General**, **Error**, **Self Service**, **Assisted Service**, and **Callback**) has a list of disposition codes that you can enable. When you select a disposition, a block for it is created below the main **Finalize** block. You can then drag other valid blocks (such as an HTTP REST block) below the disposition block to further customize the handling for that disposition.

Here's an example (click to enlarge):



When an application enters the **Finalize** phase, it has only one disposition code, so only one disposition block is executed. However, the **All** disposition code is unique in that it is always executed, in addition to (and after) any disposition block related to the actual disposition code of the application. This is the only case where more than one disposition block is executed.

Typically, you would select the **All** disposition code when you want to execute some post-processing logic, no matter what the actual application disposition code is. This is more efficient than duplicating the same logic in every possible disposition block.

Setting up handlers of the **Finalize** phase is optional. There may be no need to do anything special for these cases.

Variables

You can use two types of variables in Designer:

- User Variables - These are variables that you create. You can use these variables throughout the application and in all phases.
- System Variables - These variables are created with the application and cannot be deleted.

Tips

Variable names must be alphanumeric, but not start with a numeric character. For example:

- Valid variable names = **a**bcdef123 or **c**123badef
- Invalid variable names = **1**23abcdef or **3**abcdef21

Variable values may be:

- ECMAScript objects, such as `Date()`.
- Valid ECMAScript expressions. Do not add an ending semi-colon (;) as typically required by ECMAScript.
- Simple values, such as numeric or string.
  - If the value is a string, it must be surrounded by single quotes (for example, `'value'`). If the value also uses a single quote, you can use a backslash to escape the quote character (for example `'Joe\'s Pizza'`).

> **Important**
> The block properties page will indicate if single quotes are required.

User Variables

You can add user variables in the **Initialize** phase. You can assign initial values to these variables in the **Initialize** phase, or by setting values in an **Assign** block in the **Initialize** phase.

You can also assign a system variable as the default value of a user variable. For example, you might assign the system variable **DNIS** to a user variable you have created. (If the system variable does not have a value at the time of the call, the default values are used.) This is also supported for Self Service Shared Modules.

Important

User variables may not be initialized correctly if their value is set to one or more system variables in the **Initialize** phase itself. This phase should be used to declare variables, but their values should be assigned later using an **Assign** block if the value or the value expression involves a system variable.

Warning

You can delete a variable even if it is required by the application. Designer validates the application when you click **Publish**, at which time it checks for deleted variables.

Securing Variables

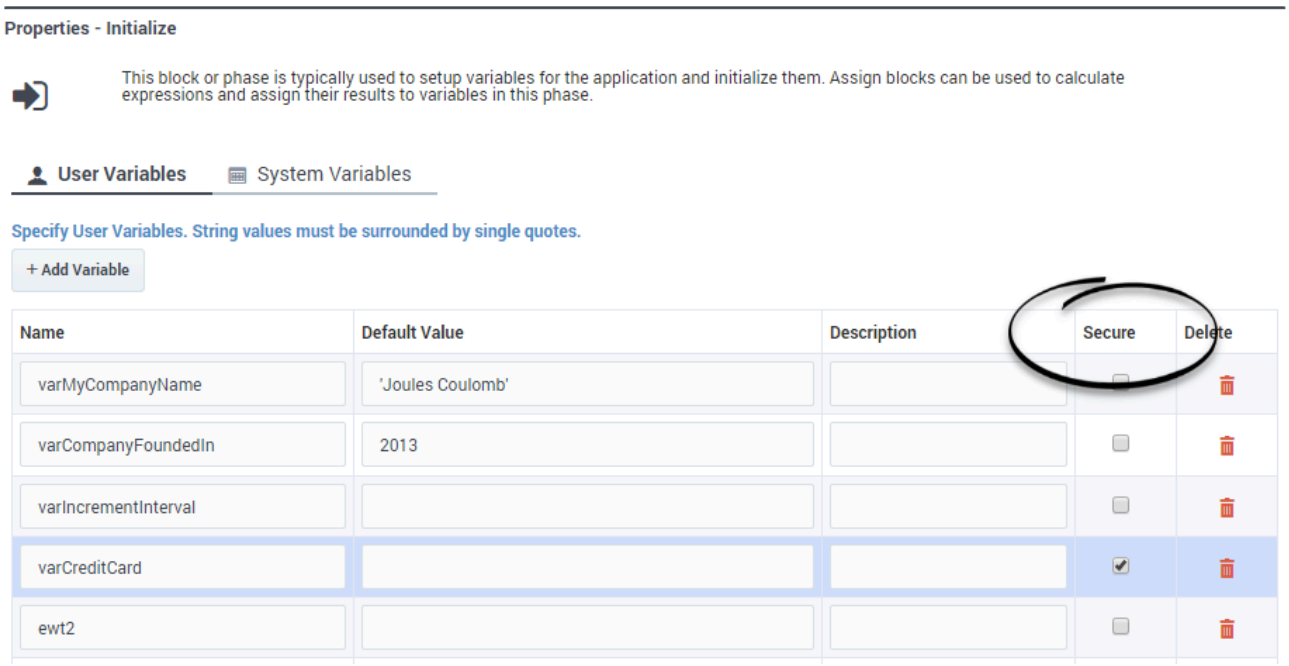
Warning

Variable values can be captured in a variety of data sources when Designer applications run on the Genesys platform. **To prevent the values of variables from being exposed as plain text in Designer and the Genesys platform, they must be marked as Secure.**

You can enable the **Secure** check box to indicate that a variable is secure and must not be logged or recorded. The application must be published for any changes to these settings to take effect.

Secure variables function as follows:

- Secure variable values are not logged in application logs.
- When you use secure variables to store the results of a user input, the user input is masked in MCP (Media Control Platform) logs.
- When you use secure variables to play back prompts, the prompt message is masked in MCP logs.
- Users cannot select secure variables for blocks that record reporting information, such as Call Data, Activity, and Milestone blocks.
- Secure variables are not reported in Analytics.



System Variables

The **Initialize** phase has a second tab that lists system variables - these variables are created with the application and cannot be deleted.

Most system variables are initialized when the application starts and can be used throughout the application, such as the **Last Milestone** variable. When your application starts, the initial value of **Last Milestone** is an empty string. While your application runs, the **Last Milestone** value is set to the last milestone that your application reaches.

The following system variables are available:

Variable Name	Description
DNIS	Specifies the dialed number.
ANI	The number associated with the calling party.
MaxTime	Maximum time (in minutes) to keep this session alive.
Timezone	The timezone used for this application, unless this value is overridden in other blocks.
Language	The default language for this application that is used for announcements.
AppLanguageName	The name of the default language for this application that is used for announcements.
RoutingSkills	A set of skills that might be specified in some blocks, such as **Menu Option** child blocks, that

Variable Name	Description
	determine how the call is routed. For example, if you select a Skill in the **Call Handling** tab of a **Menu Option** block, this selection is stored in the **RoutingSkills** variable. Then, in a subsequent **Route Call** block, you can enable the **Use system variables RoutingSkills and RoutingVirtualQueue set already in Menu Options** check box to use the value of the **RoutingSkills** variable.
RoutingVirtualQueue	A virtual queue that might be specified in some blocks, such as **Menu Option** child blocks, that is used for routing unless a different queue is specified in Routing blocks. For example, if you select a Virtual Queue in the **Call Handling** tab of a **Menu Option** block, this selection is stored in the **RoutingVirtualQueue** variable. Then, in a subsequent **Route Call** block, you can enable the **Use system variables RoutingSkills and RoutingVirtualQueue set already in Menu Options** check box to use the value of the **RoutingVirtualQueue** variable.
EstimatedWaitTime	The estimated wait time for the call to be routed to an agent.
IVRProfileName	This application's calls will be associated with the given IVRProfile for VAR reporting.
GVPTenantID	This application's calls will be associated with the given tenant for VAR reporting.
SelectedTarget	This is the DN and the switch name of the target to which the interaction was routed or should be routed to definitively. The target format is `Name@SwitchName.Type`.
SelectedVirtualQueue	The virtual queue that was selected.
SelectedComponent	The agent-level target to which the interaction was routed or should be routed to definitively. If the target selected for routing is of type **Agent**, **Place**, **Queue**, or **Routing Point**, this variable contains the target. If the desired target type is **Agent Group**, **Place Group**, or **Queue Group**, the function returns the agent, place, or queue from the corresponding group to which the interaction was sent. The target format is `Name@StatServerName.Type`.
SelectedTargetObject	This is the high-level target to which the interaction was routed or should be routed to definitively. If a skill expression is used, the function returns: `?:SkillExpression@statserver.GA` or `?GroupName:SkillExpression@statserver.GA`. The target format is `Name@StatServerName.Type`.
SelectedAgent	This is the Employee ID of the agent to which the interaction was routed.
Access	(Optional) When present, this is an ECMAScript

<table>
<tr><th>Variable Name</th><th>Description</th></tr>
<tr><td></td><td>object that represents a switch access code. The table below show its properties and the corresponding switch access code fields:<table><tr><th>Access property</th><th>Switch access code field</th></tr><tr><td>prefix</td><td>Code</td></tr><tr><td>rtype</td><td>Route Type</td></tr><tr><td>destination</td><td>Destination Source</td></tr><tr><td>location</td><td>Location Source</td></tr><tr><td>dnis</td><td>DNIS Source</td></tr></table></td></tr>
<tr><td>CustomerSegment</td><td>The segment to which the customer belongs, based on information that the customer has provided.</td></tr>
<tr><td>CustomerId</td><td>A unique identifier for the customer, based on information that the customer has provided.</td></tr>
<tr><td>EnableSSRecording</td><td>Enable call recording in the Self Service phase.</td></tr>
<tr><td>CallbackReporting</td><td>An object containing key-value pairs for callback reporting.</td></tr>
<tr><td>PositionInQueue</td><td>The call's position in queue while waiting to be routed to an agent.</td></tr>
<tr><td>AppCountry</td><td>The country code for this call (can be specified by the application).</td></tr>
<tr><td>AppRegion</td><td>The region for this call (can be specified by the application).</td></tr>
<tr><td>AppCallType</td><td>The type of call (can be specified by the application).</td></tr>
<tr><td>AppUserDisposition</td><td>A custom disposition that the application can use to specify a user-specific outcome.</td></tr>
<tr><td>AppUserDispositionCategory</td><td>A custom disposition category that the application can use to categorize user-specific outcomes.</td></tr>
<tr><td>AppDeflectionMessage</td><td>The application can use this variable to track deflections by specifying the message played when a caller disconnected their call.</td></tr>
<tr><td>AppLastMilestone</td><td>The last milestone that the application achieved.</td></tr>
<tr><td>AppStrikeoutMilestone</td><td>The last milestone that the application achieved before strikeout.</td></tr>
<tr><td>AppBailoutMilestone</td><td>The last milestone that the application achieved before the caller bailed out to an agent.</td></tr>
<tr><td>AppDeflectionMilestone</td><td>The last milestone that the application achieved before the caller was deflected.</td></tr>
<tr><td>ScriptID</td><td>The ScriptID as reported by the routing engine.</td></tr>
<tr><td>AppSelfHelpedMilestone</td><td>Used to contain a self help milestone.</td></tr>
<tr><td>SdrTraceLevel</td><td>Enables users to set the recording level. This variable accepts the following values:</td></tr>
</table>

Variable Name	Description
	• `100` — Debug level and up. Currently, there are no Debug messages. • `200` — Standard level and up. This setting shows the existing blocks array in the SDR. • `300` — Important level and up. This setting filters out all blocks except those containing data that can change from call to call (such as the **User Input** block).
AppSessionType	The type of the session. The default value is *inbound*' for inbound calls. Survey applications must use the value **survey**.
EnableRouteCallRecording	Set to `true` or `false` to enable or disable call recording for routed calls in the **Assisted Service** phase. Leave blank to use platform defaults.
GmsCallbackServiceName	The GMS Callback Service name.
GmsCallbackServiceID	The unique identifier that GMS assigns to a scheduled callback.
survey_sOffer	Set by the Setup Survey block to specify if a survey was offered, setup, or rejected.
survey_iAgentScore	Holds the user satisfaction score for the agent, if this question is asked by the survey.
survey_iCompanyScore	Holds the user satisfaction score for the company, if this question is asked by the survey.
survey_iCallScore	Holds the user satisfaction score for the overall call, if this question is asked by the survey.
survey_iProductScore	Holds the user satisfaction score for the product, if this question is asked by the survey.
survey_iRecommendScore	Holds the user's rating score (on a scale of 0-10) of the company, product, or service. Used to calculate Net Promoter Score (NPS).
ApplicationRevisionSerialID	A read-only variable that increments by 1 each time an application is revised.
ApplicationPath	The absolute path to the application.
DefaultPartition	The default partition used to provide access control in GIR. This variable can be overridden by settings in the **Record** block.
AutoStopInteraction	(Digital only) Specifies whether or not the interaction is to be automatically terminated when the session ends. The default value is `auto` (to automatically terminate a *chat* interaction if it exceeds 20 application runs or exceeds the specified expiration time); other valid settings are `true` or `false`.
ChatEntryPoint	(Digital only) Holds the point of entry for a chat interaction. Can be used in application logic at

Variable Name	Description
	runtime to provide alternative processing or to facilitate the use of parallel testing environments.
TreatmentIterationCount	Keeps track of how many times a treatment has been executed.
AgentsTotalSize	The total number of agents that are possibly available. For example, the total number of agents in a specified Agent Group.
AgentsLoginSize	The number of agents that are actually logged in.

VAR Metrics

> **Important**
>
> VAR action IDs are stripped of spaces and pipe characters (|). This includes implicit actions that are generated when a caller enters a shared module.

Use the **IVRProfileName** variable (User Data Key: **gsw-ivr-profile-name**) to associate the application VAR metrics with an IVR Profile. Use a value of `auto` to auto-detect the IVR Profile.

Use the **GVPTenantID** variable (User Data Key: **gvp-tenant-id**) to associate the application VAR metrics with a tenant. Designer attaches the value to user data. Use a value of `auto` to auto-detect the tenant.

These variables are listed in the properties of blocks once they are defined.

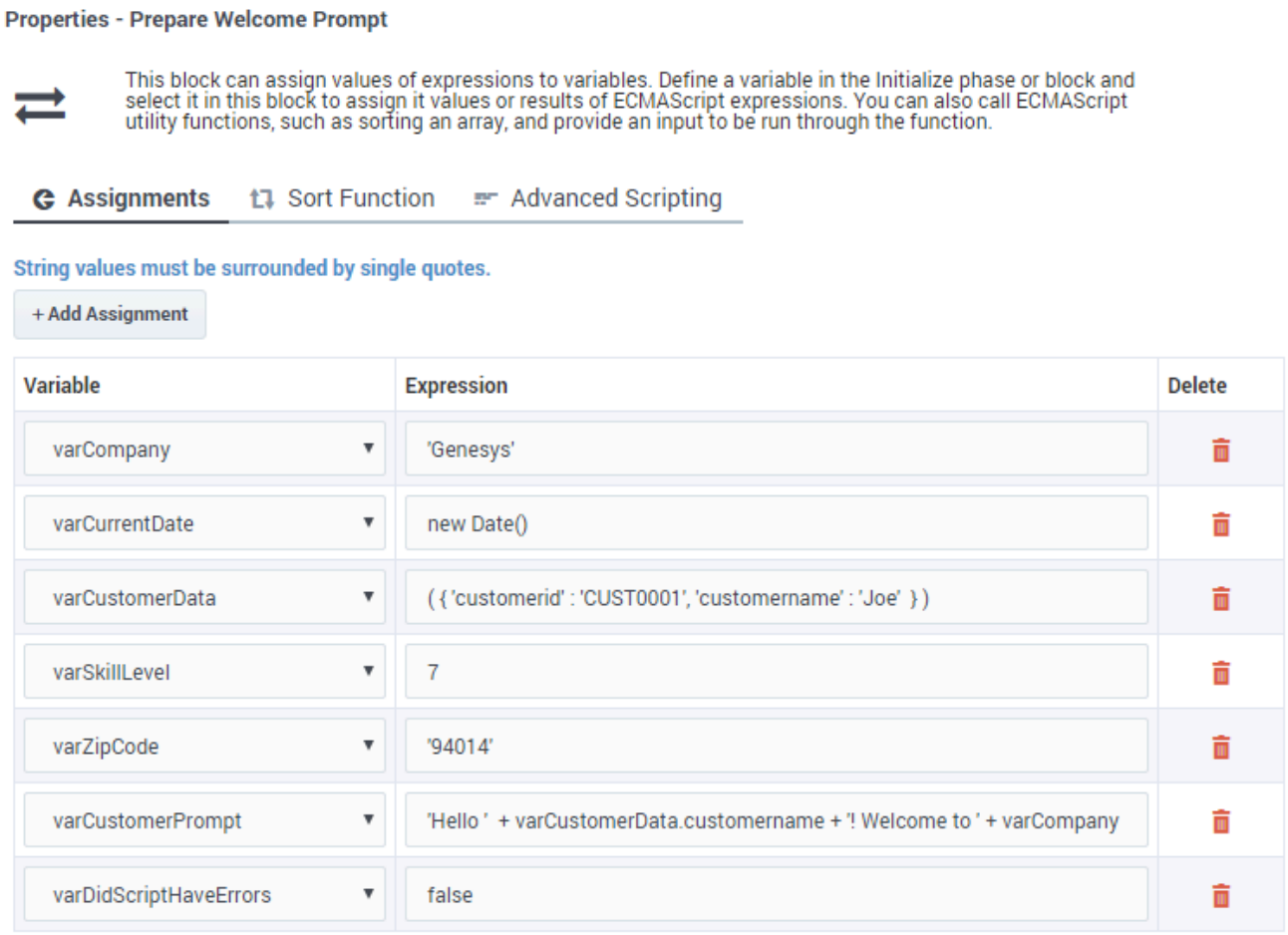
Assigning Values to Variables

Designer lets you assign values to variables in different ways. These examples show a few of the methods you can use to assign different types of values to a variable, including a JSON value.

Example 1: Simple Assignment

The easiest (and recommended) way is to assign a value to a variable using the **Assignments** tab on the Assign Variables block.

Click **Add Assignment** to add an assignment slot to the block, then choose a variable from the **Variable** column. For the **Expression**, you can use the name of another variable whose value should be copied in to the **Variable** column, a literal value (for example, "Sales Channel"), or an expression whose value should be calculated first and the results assigned to the **Variable**.



You must use single quotes (' ') when specifying string values, but you can assign numeric values without quotes. Note that the *varZipCode* above is a string data type (the single quotes tell JavaScript to treat it as a string), but it contains only numbers. It's important to remember that JavaScript treats string and numeric data types differently. For example, 1 + 2 = 3, but '1' + 2 = '12'.

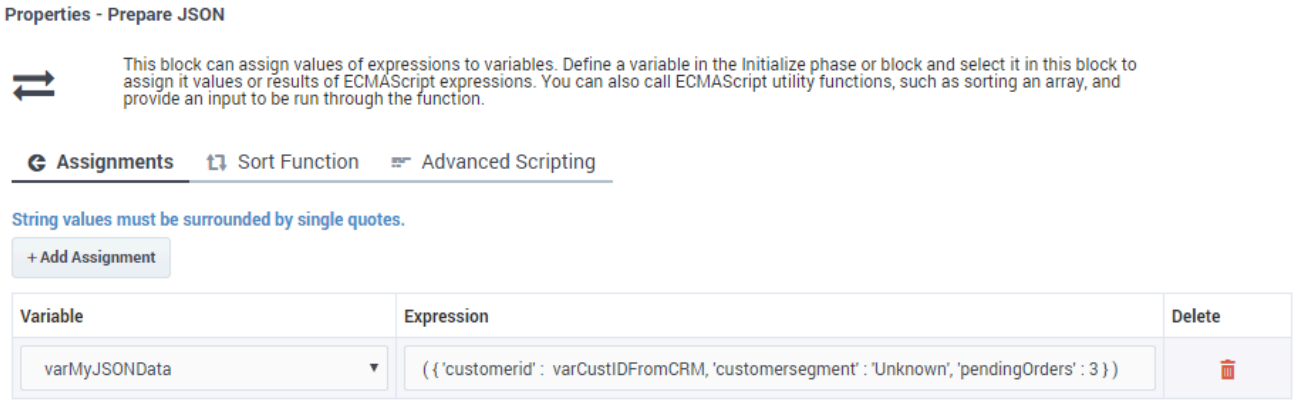
JSON data (for example, *varCustomerData*) is typically retrieved from an external data source, but you can also form a JSON string in the application. JSON strings must be enclosed in brackets ( ) and should follow the rules and syntax for JSON strings as defined by JavaScript. Note also that variables can easily be used to form part of the JSON string (as represented by *varCustIDFromCRM*, in the example shown below).

The *varCustomerPrompt* above shows a simple string expression, with the different string segments linked together by a **+**. If used in a **Play Message** block, it will play "Hello Joe! Welcome to Genesys." It accesses a property of the *varCustomerData* object using the "." notation and combines it with the welcome message.

> **Important**

> Although the terms ECMAScript and JavaScript are used interchangeably throughout this Help, Designer technically supports ECMAScript and does not support JavaScript functions that are typically used for web-browser based applications, such as pop-up windows, alerts, and so on.

Here is another example of how you could build a JSON expression. It contains mostly hard-coded strings, but also uses a variable to form part of the JSON string:



Example 2: Advanced Scripting

If your application requires you to go beyond simple assignments and use JavaScript constructs like loops or multiple nested conditions, you can use the **Advanced Scripting** tab, which allows you to compose valid ECMAScript or JavaScript.

> **Important**
>
> Advanced Scripting is an optional feature and might not be enabled on your system. To enable this functionality, contact Genesys.

To use this feature, you need a basic level of familiarity and understanding of ECMAScript syntax and rules. Any errors in the script can cause erratic behavior, so test your changes to make sure that your script works correctly.

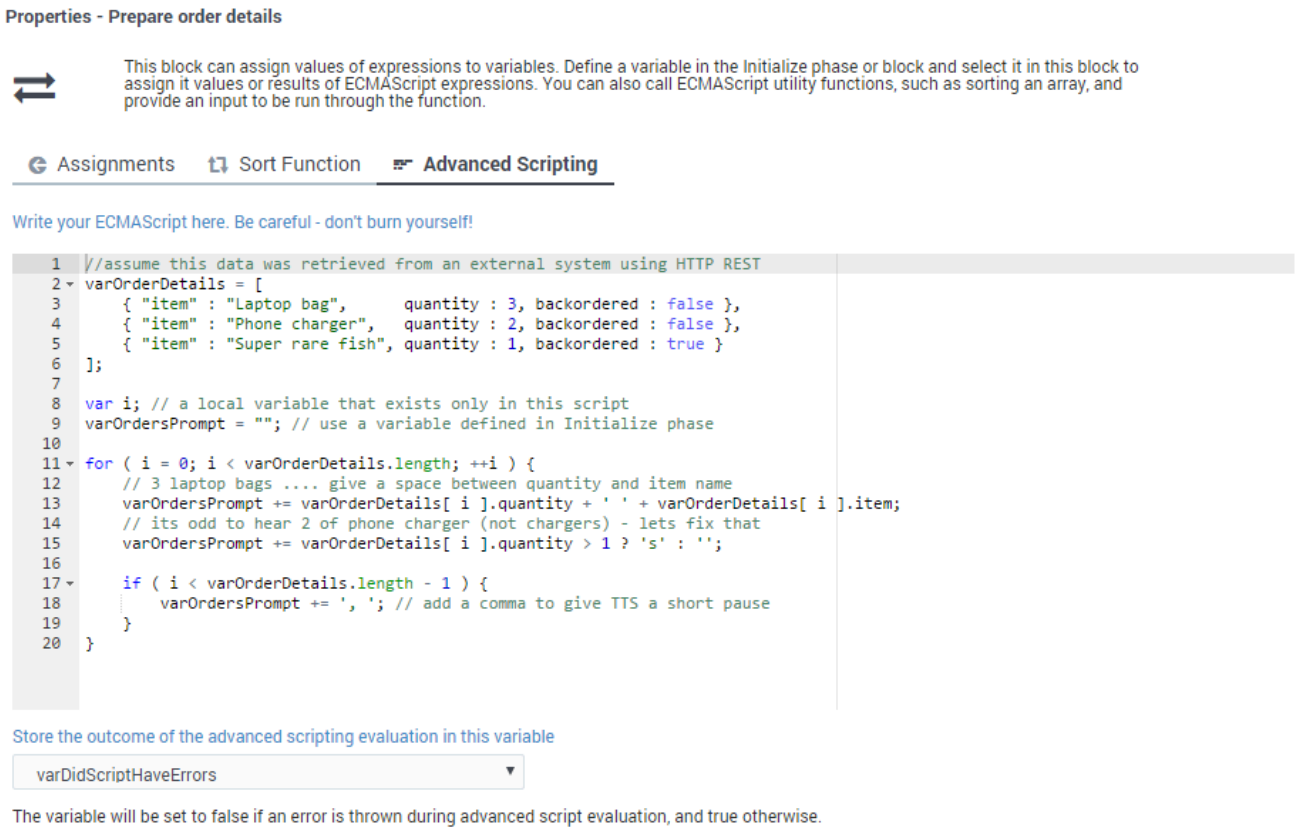
> **Warning**
>
> Use caution! Designer can check your script for syntax errors, but cannot validate it nor check for runtime errors that might occur when the script runs during a call.

In this example, the script sets the variable *varOrdersPrompt* to "3 Laptop bags, 2 Phone chargers, 1 Super rare fish". Here's how it works:

The sample script below first initializes JSON data in `varOrderDetails` so that it becomes an array of three JSON objects. Each JSON object has properties — item, quantity, backordered. The script then proceeds to loop through orders and forms a string to play back to the caller to notify them of their order status.

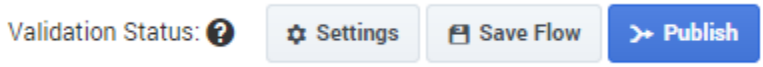
Note that this script uses variables in two scopes:

- A scope exclusive or local to this script itself ("`i`"). This variable remains available only while this script runs, and then it disappears.
- Top level variables that were defined in the **Initialize** phase — these remain available throughout this application flow, but not in any modules this application calls (such as `varOrdersPrompt`).



Saving and Publishing Your Application

It is a good idea to manually save your work often, especially after you have made important changes. If you forget to save, Designer periodically saves a temporary backup of your work.



Click **Save Flow** to save your application. This action saves your work and performs some validation checks on your application. If no problems are found, a green check mark appears beside the **Validation Status** field. Otherwise, if problems are found, a warning icon appears beside the **Validation Status** field. You can click the warning icon to display the list of warnings.

When you are ready to test and deploy your application, click **Publish**. Designer performs another validation test on your application and, if no errors are found, it generates the code that will run on Genesys platforms.

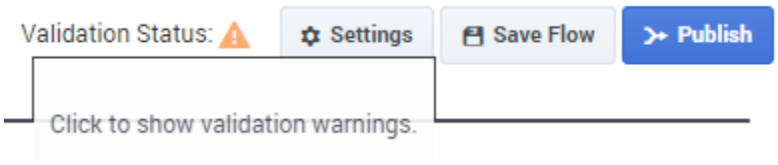
Important

Designer saves your latest edits if you are logged out due to inactivity. After you log in again and open the application, Designer displays a prompt to indicate it has found a local backup of your application, along with a comparison of timestamps between this local backup version and the version that is saved on the server. In this prompt, you can click one of the following buttons:

- **Last Manual Save**—Designer discards the local backup and opens the server version of the application. The discarded backup version cannot be recovered again.
- **Recover Backup**—Designer recovers the local backup version. You can then choose to click **Save Flow** to save your changes to the server.

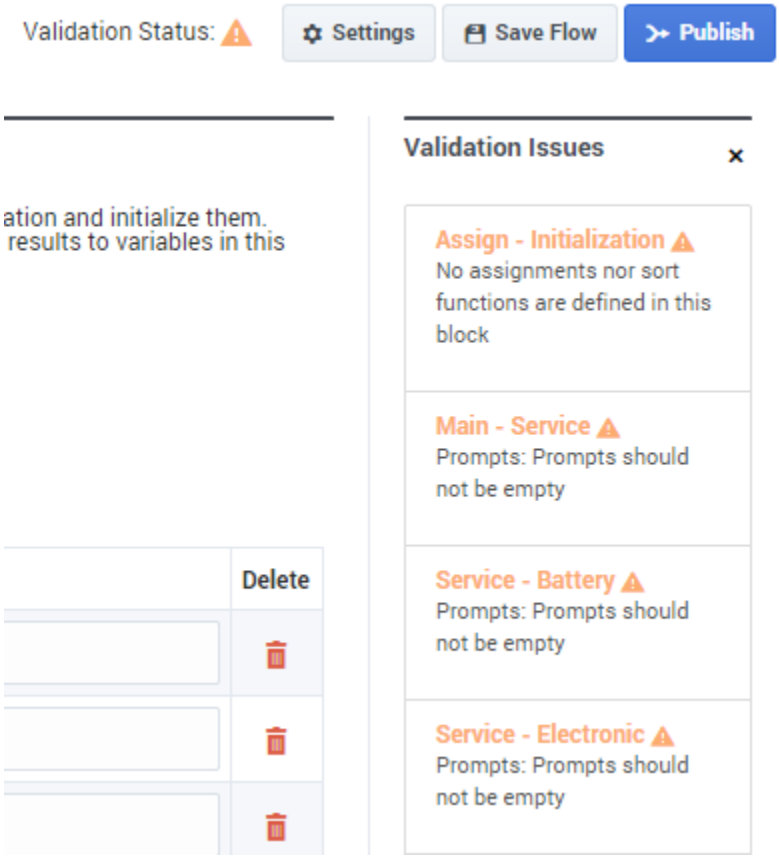
Validation Issues

If errors are found in your application, you can click the red exclamation icon beside the **Validation Status** field to display the **Validation Issues** list.



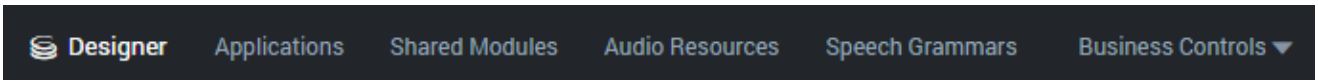
The **Validation Issues** list displays warnings in yellow and errors in red. Code generation can complete if warnings are present, but fails if errors exist.

Click a warning or error to return to the block containing the issue and address the problem.



Navigation Bar

The Navigation Bar is always present on your screen. It provides one-click access to applications, shared modules, audio resources, speech grammars, and business control objects.



Click a topic below to learn more:

- Applications - Read this page to learn more about topics such as how to clone an application, how to assign a phone number to an application, and how to enable or disable an application.
- Shared Modules - Read this page to learn more about topics such as which types of shared modules are available and how to version shared modules.
- Audio Resources - Read this page to learn more about topics such as how to manage audio resources and how to incorporate audio resources in your applications and shared modules.
- Speech Grammars - Read this page to learn more about speech grammars and topics such as how you can upload speech grammars and how you can use speech grammars in your applications.
- Business Controls - Read this page to learn more about topics such as emergency flags, business hours, special days, and data tables.

Applications

Click **Applications** in the navigation bar to manage your applications. The Applications page enables you to add a new application, clone an application, or delete an application. You can also create tags to better organize your applications.

When your application is ready, you can assign a phone number and enable your application to receive calls.

Creating an Application

You can create an application by clicking **Add Application**. In the **Creating new application** pop-up window, enter a name in the **Name** field. Depending on your configuration, you might be able to choose an application type in the **Type** field. The following types are available:

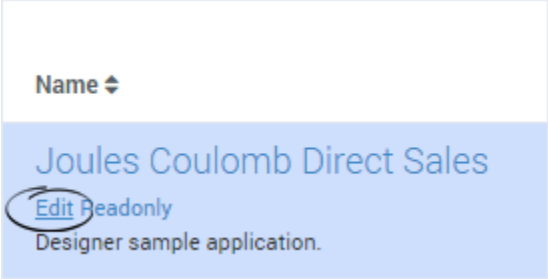
- **Default** - Choose this option to create an Orchestration application. This type of application can have **Self Service** and **Assisted Service** phases, in addition to the **Initialize** and **Finalize** phases. This type supports Voice calls only .
- **Digital** - Choose this option to create an application that can process multichannel digital interactions (no voice).
- **Callback** - Choose this option to create an application that is only used for outbound callback calls. See the Callback blocks page for more information.
- **IVR** - Choose this option to create a VoiceXML-only application that can use future functionality that will only be available to this application type, such as Cisco ICM support. This type of application is similar to **Default**, but it does not have an **Assisted Service** phase and cannot specify any call-routing logic. In addition, the following blocks are not available in **IVR** type applications:
  - Route Call
  - Statistic
  - Transaction List
  - Query VQs

> **Important**
> You cannot assign a phone number to **IVR** type applications.

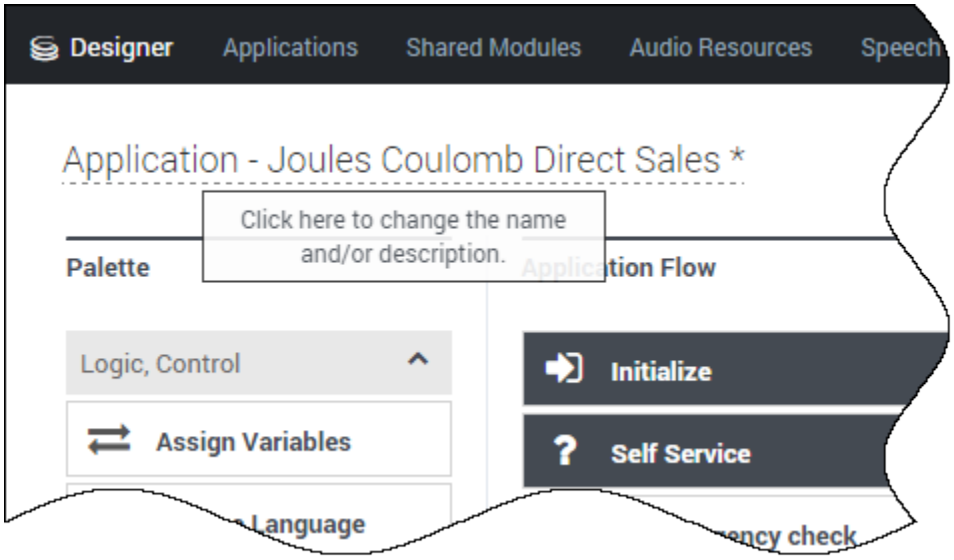
Changing Application Properties

Rename an Application

In the Applications list, hover over an application name and click **Edit**.



Alternately, you can also edit the application name and description by clicking the application name when it is open for editing. Hovering over the name shows you the current description.



Last application snapshot

> **Important**
> This feature is only available in Designer versions 8.5.202.82 and later.

When applications are published, Designer compiles them into executable code and creates a unique "snapshot" for each application. The snapshot preserves the executable code for an application as it was at the time the snapshot was taken.

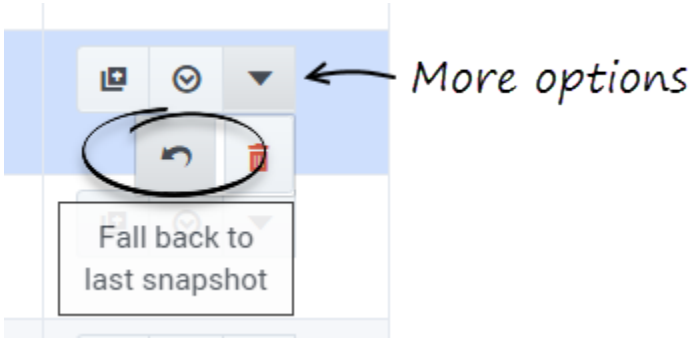
> **Important**
> Changes to associated resources (such as audio, data tables, and business controls) don't require the application to be published again, so they do not affect the snapshot.

After an upgrade, Designer uses the last snapshot of an application to minimize the amount of changes introduced to the environment as part of the upgrade.

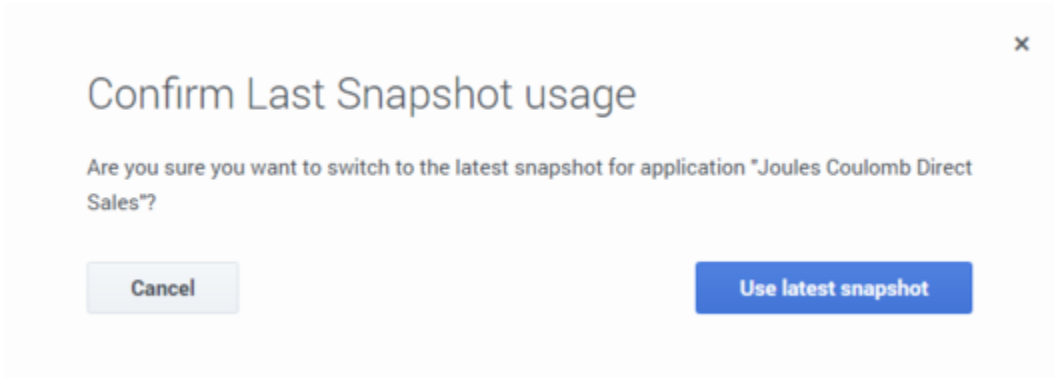
Designer continues to use the last application snapshot until the application is manually published again, at which point Designer deactivates the snapshot and then compiles and activates the new executable code.

The snapshot is retained and stored as part of the application properties. If there are issues with the newly published application, you can reactivate the snapshot by using the **Fall back to last snapshot** option, which allows you to restore live traffic handling while troubleshooting the problem.

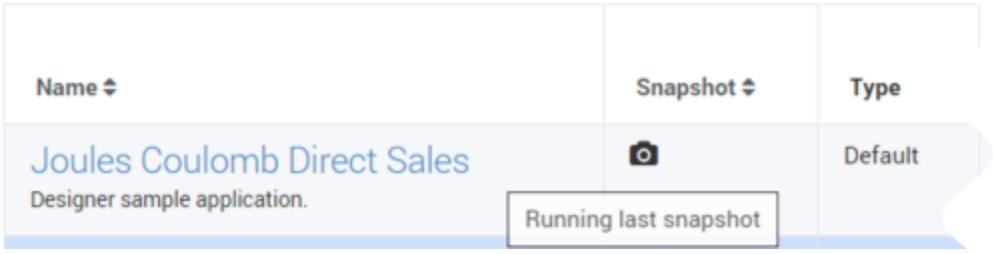
To do this, select **Fall back to last snapshot** from the **Actions** column (you might have to select **More Options** to see this option):



Confirm that Designer should use the last application snapshot:



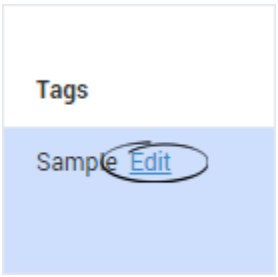
If an application is running the last snapshot, a camera icon appears in the **Snapshot** column:



Tag an Application

You can assign a tag to multiple applications so they can be collected into groups and be searched.

Hover over the **Tags** column for an application and click **Edit**.

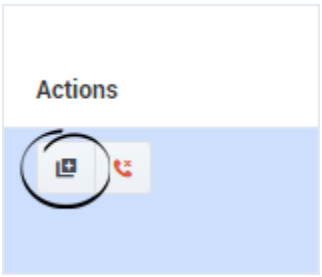


To delete a tag that is longer than the column width, position the cursor after the tag and use the backspace key to delete it.

Clone an Application

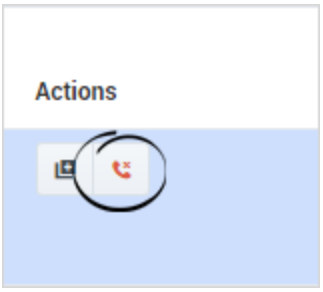
You can clone (copy) an application. This is useful if you want to test changes but you do not want to disrupt routing strategies that use the existing application.

In the **Actions** column, click the **Clone** icon.



Disconnect Phone Number(s)

To disconnect all phone numbers from an application, click the **Disconnect Phone Number(s)** icon. Once disconnected, the numbers can be assigned to other applications.



If you want to disconnect specific numbers from an application, click **Manage** and deselect the numbers to disconnect.

Managing Application Settings

Click **Settings** in an application to access its settings.

General Tab

- **Application Reporting Title** - Specify the name of this application that will be used in reporting.
- **Application Version** - Specify a version number for this application.
- **Stage** - Select an application stage. If resource caching is enabled, the application stage also manages how often cached resources are checked for updates. By default, applications are in the **Live** stage.

Audio Tab

- **Audio Resource Collection** - Select the Audio Collection that this application will use. See the Media Resources page for more information.

Reporting Tab

- **Milestone Path Prefix** - Specify a prefix to use with this application's milestone paths.

DTMF Options

This tab enables you to set global DTMF commands for your application. These DTMF keys can be used at any time within the application to trigger a specified action.

A common use case for this feature is a global command for the DTMF key **0** that routes the caller directly to an agent. In this example, you can set **0** as a global DTMF command that routes directly to the **Assisted Service** phase. In your application, you can add a Play Message block to announce that callers can press **0** at any time to speak to an agent.

Selecting **Enable Global Commands** enables global DTMF commands for the application.

To set a global DTMF command, select the drop-down menu beside the corresponding DTMF key that you want to use. In the drop-down menu, select a target block or phase for the DTMF key. Click **OK** when you are done setting global DTMF commands.

Global DTMF commands can target the **Self Service**, **Assisted Service**, or **Finalize** phase, or any block within the **Self Service** phase.

> ### Important
>
> - If the same DTMF key is also used by a block within your application, Designer first processes the command in the block.
> - You can also use global DTMF commands with **Self Service** type shared modules.

Speech Recognition Tab

Configure settings for speech recognition (ASR). See the User Input block page for more information.

Global Retry Tab

Configure global settings for menu retries. See the Menu block page for more information.

Caching Tab

Resource caching can improve overall system performance, but it can also cause a delay in how long it takes for changes in Designer to take effect (changes to Data Tables and Business Controls take effect immediately).

In most cases, it is best to leave **Use workspace cache settings** selected (it is already selected by default), as each application stage has optimized settings for how often a resource is checked. But you can disable it if you want the application to regularly check the cached resources for updates, and enter your own values for each resource.

The following resources are cached:

- Media Resources
- Speech Grammars

> ### Important
>
> Default values are set by Genesys for optimal performance. Although you can change these values, doing so might negatively impact application performance. Contact your Genesys representative for additional information that might apply to your environment.

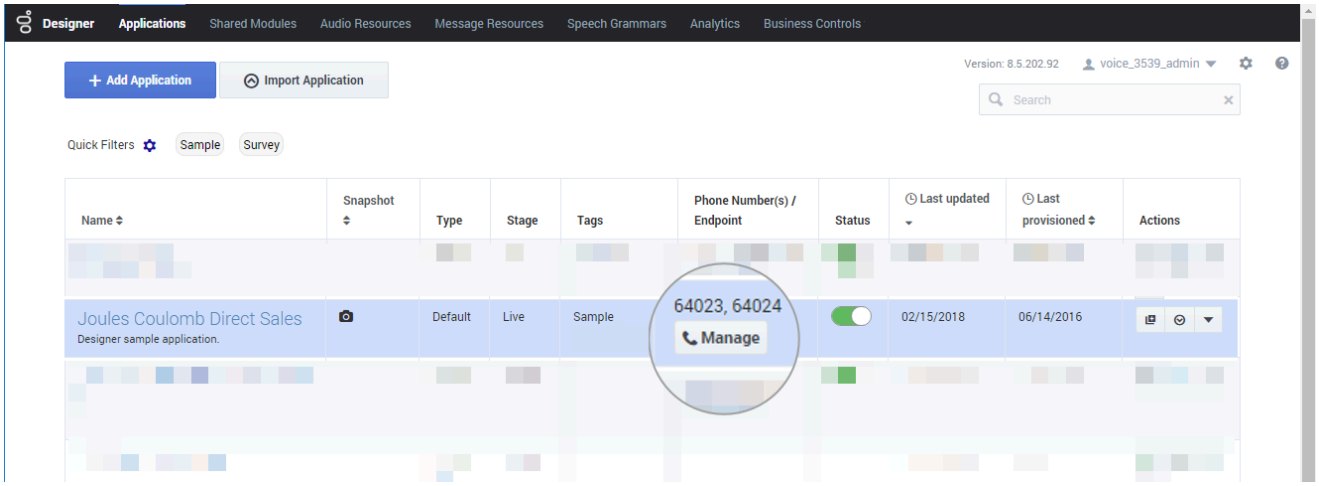
Misc Tab

Enabling the **Tracing** option enables additional data to be collected while the application runs, which can later be used for debugging.

> **Important**
> This option should only be enabled when required, as it impacts application performance.

Assigning a Phone Number or Endpoint

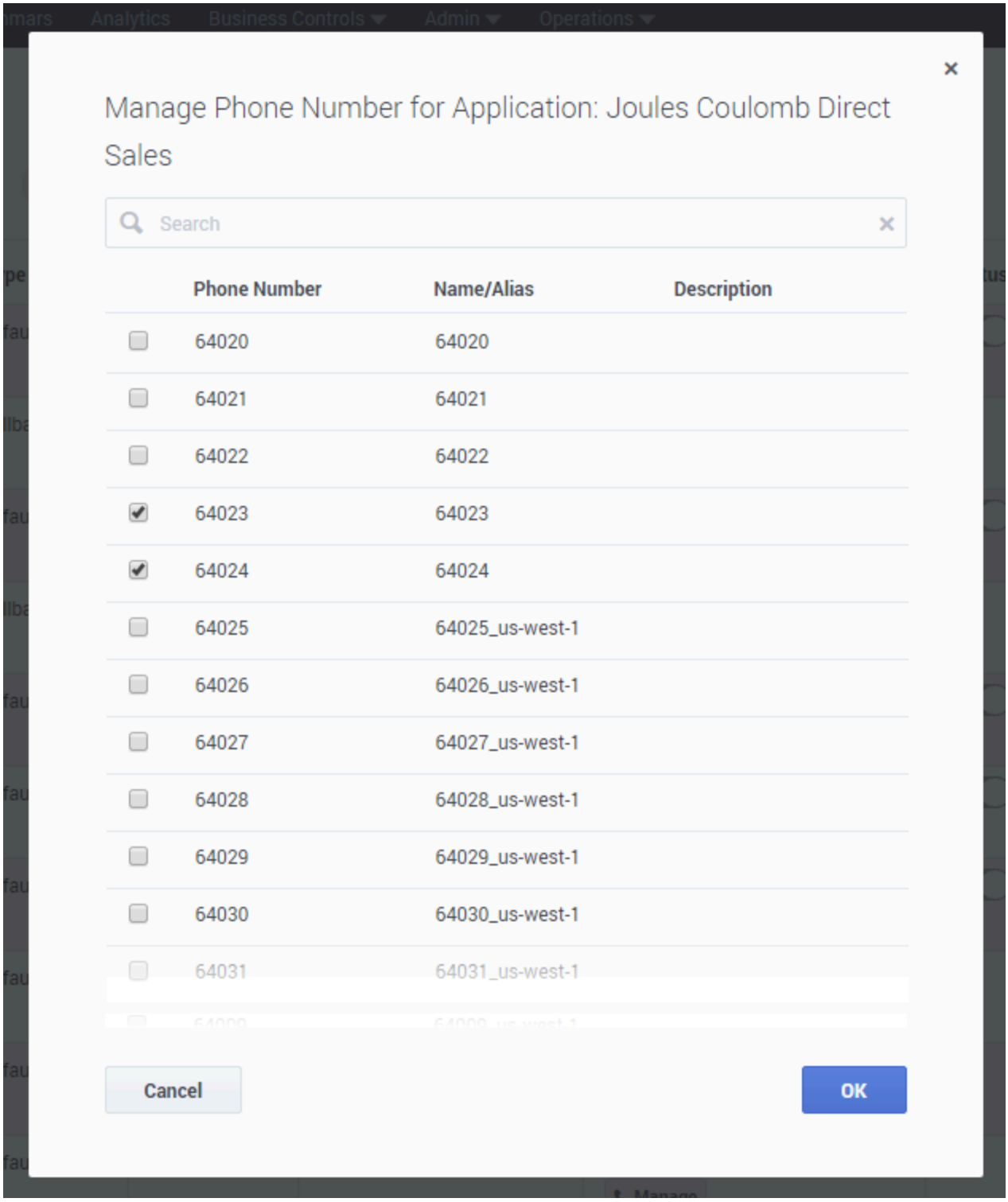
Click **Manage** in the **Phone Number(s)/Endpoint** column to select and assign a phone number to the application. If the application is a digital type, you can use this setting to assign a chat endpoint.



Assigning a phone number

- Select the phone number(s) that you want to assign to the application.

Example


Analytics
Business Controls
Admin
Operations
Manage Phone Number for Application: Joules Coulomb Direct Sales
Search
Phone Number
Name/Alias
Description
64020
64020
64021
64021
64022
64022
64023
64023
64024
64024
64025
64025_us-west-1
64026
64026_us-west-1
64027
64027_us-west-1
64028
64028_us-west-1
64029
64029_us-west-1
64030
64030_us-west-1
64031
64031_us-west-1
Cancel
OK

Important

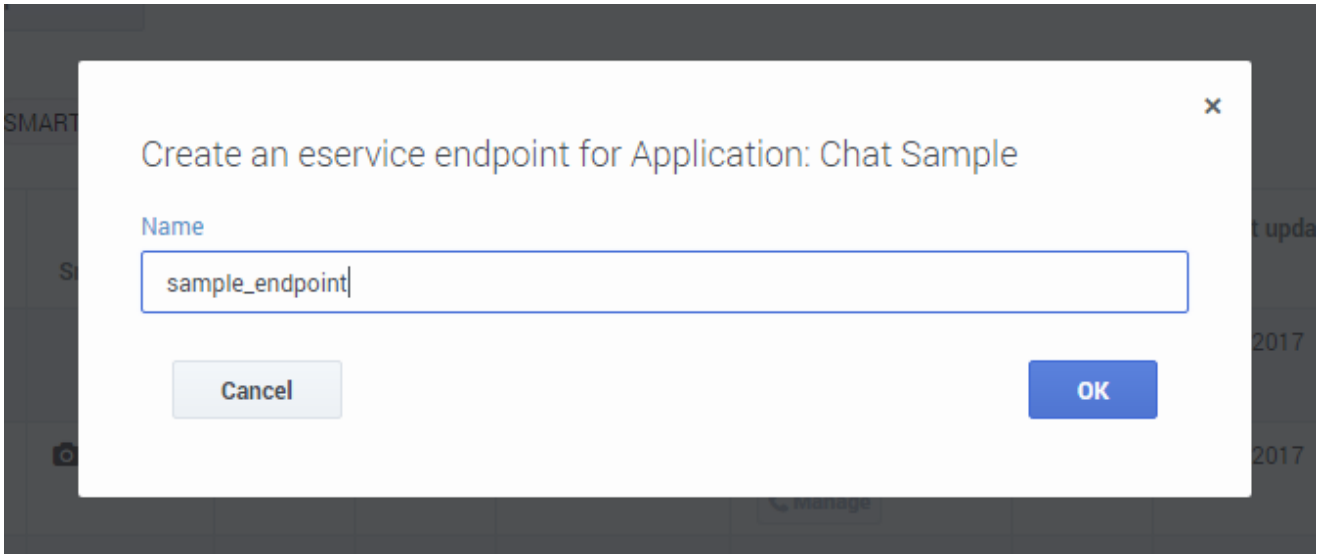
- You cannot assign a phone number to **IVR** type applications.
- You cannot assign a phone number to an application that has not been published at least once.
- You might see a warning symbol beside a phone number that is assigned to your application. This warning symbol could indicate:
  - The phone number was not assigned by using Designer.
  - The application was updated but the changes were not published.

    This warning symbol indicates there could be a problem with the phone number - it does not indicate whether the phone number is functional. You can ignore the warning if you are certain the phone number is functional. If you are not sure whether the phone number is functional, contact your Genesys representative.

Assigning a chat endpoint to a digital application

- Enter a name that is unique across all applications.
- The name should indicate the origination point of the chat (for example, *sales_page* or *mortgage_division*).
- Use alphanumeric characters only. Avoid using spaces or special characters (underscores are okay).

Example



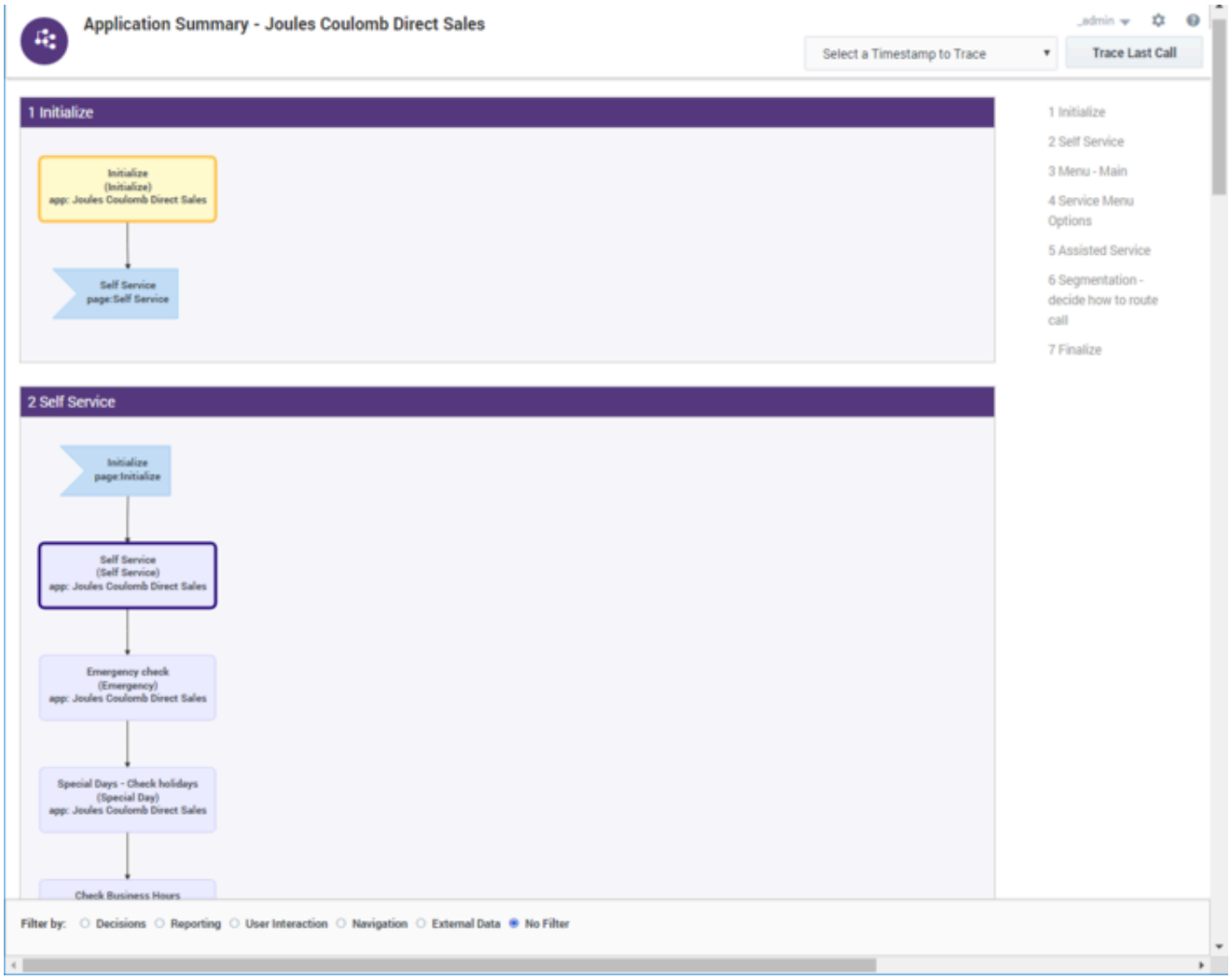
Enabling Your Application

After you have assigned a phone number to your application, you can enable it by clicking the switch icon in the **Status** column. The switch icon turns green when the application is enabled.

Viewing the application summary

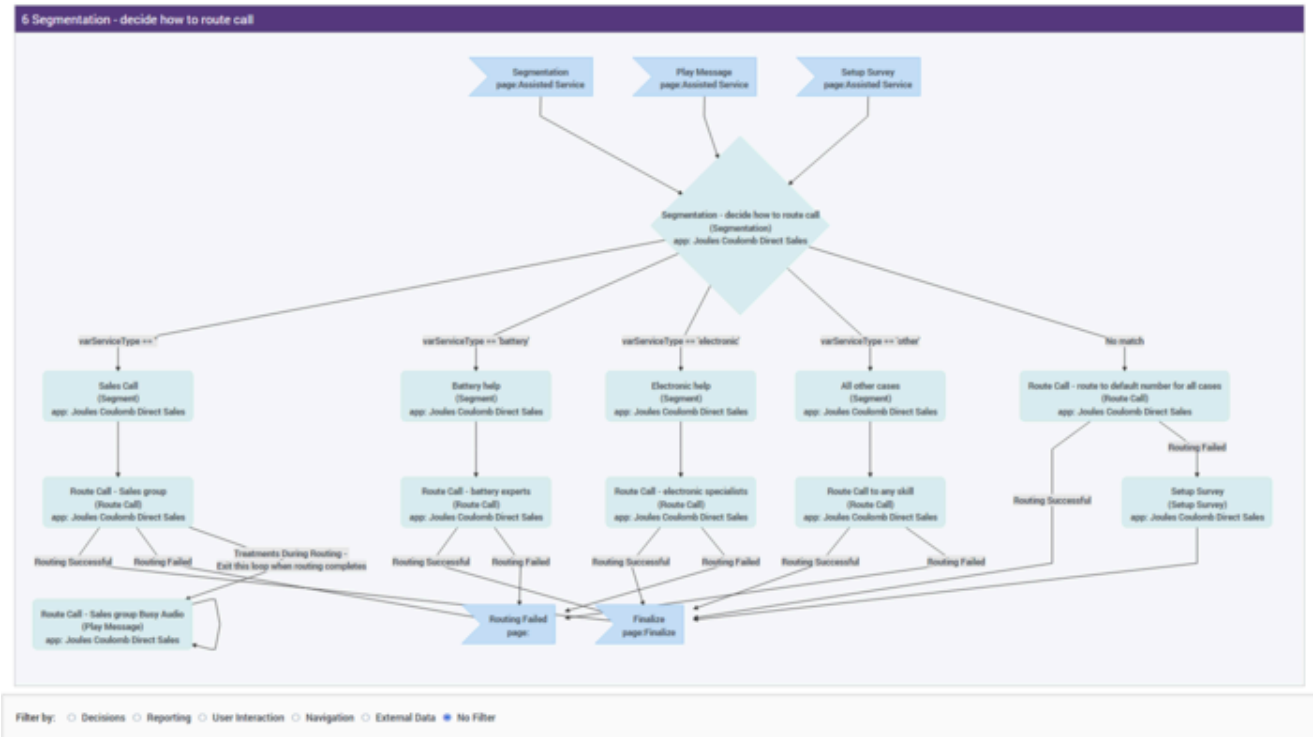
When an application is displayed the flow section, you can use **Views** > **Summary** to generate a visualization of the application.

The resulting diagram is similar to a hierarchical flow-chart, with each node representing a block in the application flow:



The application summary view shows all the possible paths that an interaction can take through the application. The diagram is divided into sections for each application phase, and for nodes that need to be expanded into their own sections due to their size or complexity.

For example, this section shows a **Segmentation** node:



Filtering

Use the **Filter by** options to focus on specific details. You can choose to filter the diagram by **Decisions**, **Reporting**, **User Interaction**, **Navigation**, or **External Data**.

Select **No Filter** to clear any selected filtering options.

Session playback

You can use **Select a Timestamp to Trace** to select and display the path that a specific session took through the application, or click **Trace Last Call** to load the path of the last session that was processed.

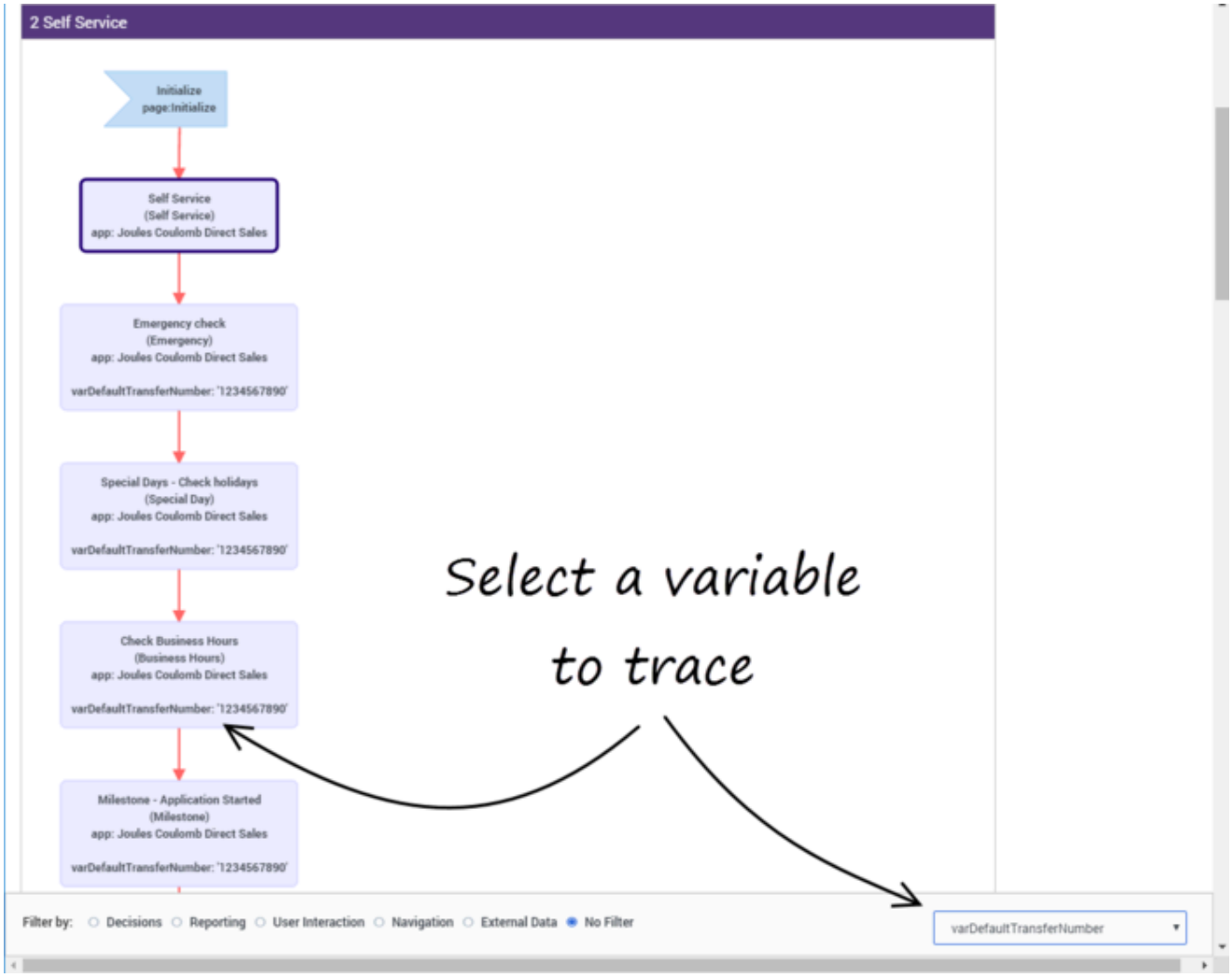
The path that the session took is indicated by red flashing connecting lines and highlighted nodes:

Des apps summary trace.gif

Click **Reset** to clear the playback details for the selected session.

Variable tracing

Likewise, you can also select a specific variable to trace:



This lets you track a variable as it moves and changes through the various nodes, which can be useful for discovering and resolving potential trouble spots.

Shared Modules

Click **Shared Modules** in the navigation bar to manage your shared modules. Shared modules are small pieces of applications that you can use in one or more applications. If you change a shared module, you are also changing all of the applications that use that module.

There are four types of shared modules:

- Self Service - Used within the **Self Service** phase of an application.
- Assisted Service - Used within the **Assisted Service** phase of an application, or within the **Initialize** phase with certain restrictions (see the note below).
- **Templates** - Used with the Callback block. These templates are read-only and cannot be edited or deleted. Click **Hide Templates** to hide these shared modules in the list.
- **Digital** - Used within Digital application types.

> **Important**
>
> - You cannot switch the type after you create the shared module.
> - Before you publish a module, you must ensure the module is adequately developed for use within applications. It should have well-defined input and output parameters that can be specified in the host application.
> - The **Route Agent**, **Route Call**, **Voice Mail**, and **Callback** blocks are not supported in **Assisted Service** type shared modules if the shared module is used in the **Initialize** phase.

Using Shared Modules

Modules can be used in different ways, depending on how you are planning your application. Currently, you can use modules with the following blocks:

Shared Module blocks

Larger application flows can often be difficult to manage. By dividing a larger application into smaller segments of individual flows, you can then "stitch" it back together using Shared Module blocks. This helps to make the flow size more manageable and also promotes reuse within and across applications.

Routing blocks (as "Busy Treatments")

You can use **Self Service** type modules to play *busy treatments* for callers (for example, play them

some music while they wait to be connected with an agent). They can also be used to keep callers updated about their estimated wait times and, if the times seem excessive, offer them additional services (such as Callback). These modules will loop automatically until the call is routed, the caller hangs up, or the timeout specified in the routing block expires -- at which point the next block in the application is triggered.

- Learn more about routing blocks and busy treatments

Start Treatment block

The Start Treatment block also offers busy treatments to callers, but works a bit differently than the ones offered by routing blocks. Typically, you would use this block in the **Assisted Service** phase when you want to start a busy treatment (for example, play an audio file to callers while they wait to speak with an agent) and then move on to the routing blocks, without interrupting the playback to the caller.

- Learn more about the Start Treatment block and busy treatments

Creating a Shared Module

Click **Add Module** to create a new module. At this stage, it does not have a version number associated with it and it is not visible to applications. In the pop-up window, enter a name for the module in the **Name** field and specify in the **Type** drop-down whether this module is for the **Self Service** or **Assisted Service** phase.

When you are editing a module, you can perform the following actions:

- **Save Flow** - Save and validate changes to the module. Each save creates an incremented revision.
- **Create Version** - Create a new version of the module and make it available to applications. You must specify the following information:
  - **Version label** - The version number (for example, **1.0**).
  - **Notes** - Relevant information for use of this module.

Importing a Shared Module

You can import a shared module that was previously exported from another Designer workspace by clicking **Import Module**. Click **Choose file** to browse to the location of the archived file, then click **Upload**.

If everything looks ok, click **Confirm**. Designer will import all versions of the module that were exported to the archived file and upgrade any versions that need to be upgraded.

Settings

Click **Settings** in a Shared Module to access its settings.

General Tab

Managing Audio Resources for Shared Modules

In the **Audio Resource Collection** drop-down menu, select **Inherits Audio Collection from Calling Context** if you want the shared module to inherit its audio collection from the host application. Otherwise, select an audio collection in the drop-down list if you want the shared module to only use a specific audio collection.

Example

You have one shared module that is used within two applications.

First, open the shared module and click **Settings**. In the pop-up window, select the **Audio Resource Collection** drop-down menu and choose **Inherits Audio Collection from Calling Context**.

Next, in the module's **Play Message** block, specify the type is **Announcement** and ensure the **Variable?** check box is checked. Choose a variable name (this example uses **greeting**).

Finally, in each application's Audio Collections, create an announcement called `greeting`. When playing the **Play Message** block, the call searches the inherited calling context audio collection for **greeting**.

Milestone Path Prefix

Specify a prefix to use with this application's milestone paths.

DTMF Options Tab

> **Important**
> This tab is only available for **Self Service** type shared modules.

This tab enables you to set global DTMF commands for your shared module. These DTMF keys can be used at any time within the shared module to trigger a specified action.

To set a global DTMF command, select the drop-down menu beside the corresponding DTMF key that you want to use. In the drop-down menu, select a target block for the DTMF key. Click **OK** when you are done setting global DTMF commands.

> **Tip**
> You can also set global DTMF options for Applications. In this case, when the shared module is running, global DTMF options are first processed within the shared module and then within the application.

Global DTMF Options Among Shared Modules

Your shared module (*Shared Module A*) might interact with another shared module (*Shared Module B*) that also has global DTMF commands. In this case, Designer processes DTMF commands in this order:

1. By block. For example, a DTMF command within a **User Input** block or a **Menu** block that expects this DTMF command.
2. By global setting in *Shared Module B*.
3. By global setting in *Shared Module A*.
4. By global setting in the host application.

If the DTMF command is not used by one of the above, Designer discards the command.

Other Actions

The following additional operations are available in the **Actions** column:

- **List module versions** - Lists the available versions.
- **List module consumers** - Lists applications that use any version of this module.

> **Important**
> It is critical that you review the list of applications that use a module before the module is deleted. A module should not be deleted if it is used by an application.

- **Clone module** - Clone the selected module and save it with a new name.

> **Important**
> A cloned module does not inherit the history and published versions of the original module.

- **Export module** - Export the selected module for use in another Designer workspace. When you export a module, all versions of that module are exported, including the unpublished version.

> **Tip**
>
> If you are using a Safari browser to export a module, the exported file is downloaded as *unknown*. The file is valid and can be imported successfully, but you might want to rename it to something more meaningful.

- **Delete module** - Deletes all versions of this module. *Published* applications that already use the module (in other words, applications that have already generated their code) are not affected.

Media Resources

Click **Media Resources** in the navigation bar to manage your media resources.

Each **Media Collection** contains a set of **Media Resources**, which are TTS (Text-to-Speech) prompts and recorded audio files that you can use in your applications and shared modules in any block that supports playing audio.

For example, a banking company might have a media collection assigned to an application that receives customer inquiries. Within that collection are several media resources for each language in which the company operates. Each media resource contains language-specific audio for greeting the caller, communicating announcements, and offering special promotions.

From this page, you can centrally manage these media resources for all of your applications. If you change a media file or TTS text, the change takes effect the next time that the media collection is published.

See the Applications or Shared Modules page for more information on how to assign a media collection to an application or shared module.

Creating a media collection

To create a new media collection, click **Add Media Collection** and enter a name. When you are done, click **Create and Open** to open the new collection and add media resources.

To specify a media collection for an application or a shared module, go to the **Media** tab in the application settings and select the **Media Resource Collection** you want to use. Each application or shared module can be linked to a single media collection. If you don't specify a collection for a shared module, it inherits the collection of the invoking application.

To improve media resources management, consider linking shared modules to their own media collections.

Tip

A media collection can't have more than 5000 prompts—if you need more than that for an application, organize the prompts into multiple collections. You can then split your application into modules, with each module referencing the appropriate collection.

- Keep this limit in mind when planning your applications. It is much more difficult to go back and split an application later if this limit is reached.
- Splitting an application into multiple shared modules lets you use multiple media collections in the application (each shared module can work with only one user-defined media collection).

- **Important**: Designer won't warn you or display an error if there are too many prompts. But as more prompts are added, and the limit is reached or exceeded, your customers might start to experience quality issues due to the additional processing required.

Adding media resources

Click **Add Media Resource** and enter a name for the media resource. Make sure to use a unique name as you won't be able to add it if it has the same name as an existing system resource. Click **OK**.

Properties of the new media resource are displayed to the right of the list. You can choose to add a **Description** to describe the purpose of this resource. You can also add **Tags** to relate this resource to similar media resources with the same tags.

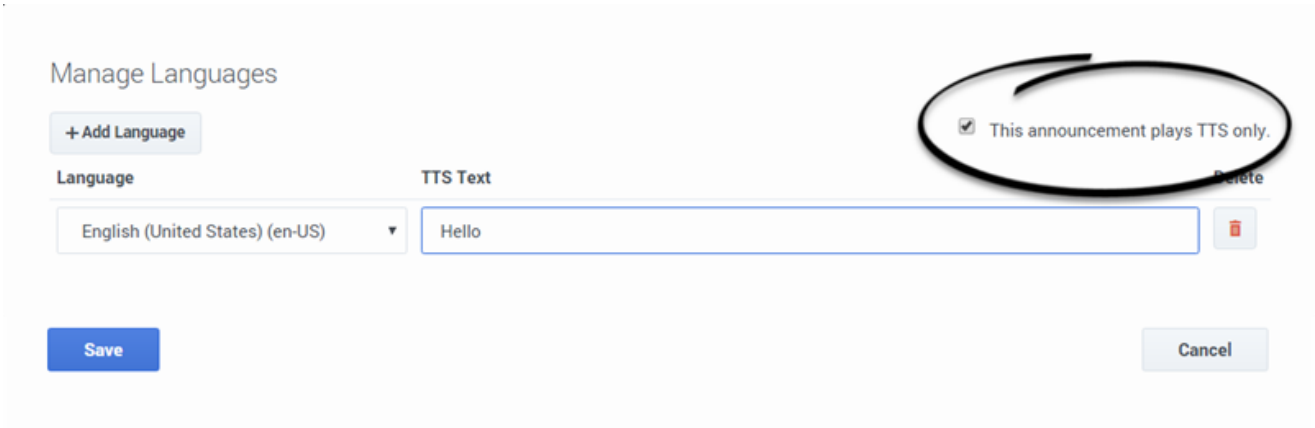
Click **Manage Languages** to continue adding the media resource. You can add two types of media resources: TTS (Text-to-Speech), where you provide the words that the computer will speak, and audio files, where you upload a file that contains an audio recording.

Important

- Media files should be named as follows: *<media resource name>_<short language name>.<file extension>*. Example: welcome_en-US.wav. The file name must only contain alphanumeric characters, underscores (_), hyphens (-), or dots (.). File names with special characters are not supported.
- Audio files must be recorded in (or converted to) the G711 u-Law 8 kHz (also known as G.711Mu or G.711µ) audio codec. They must also be single-channel (mono) and saved as a .wav file.

Adding a TTS resource

- Click **Add language** to select the language to be spoken.
- Select **This announcement plays TTS only**.
- In the **TTS Text** field, enter the text to be spoken by the TTS engine and **Save**.



Adding an audio file resource

- Ensure **This announcement plays TTS only** is NOT selected.
- Click **Add language** to select the language spoken by the audio resource.
- Click the **File** button to select the audio file to upload. A checkmark appears if the file is uploaded successfully.
- (Optional) You can enter **Alternate TTS Text** to be spoken if the audio file is unable to play.
- After the audio file is uploaded and you have saved the audio resource, you can click the play button to play the audio file.

Publishing a media collection

Media collections must be published before they can be used by applications. When you are finished adding media resources, click **Publish Media Collection** to publish the collection. If you want to undo the changes you have made, click **Revert Changes**. This will discard any changes made and

restore the collection back to the last published version.

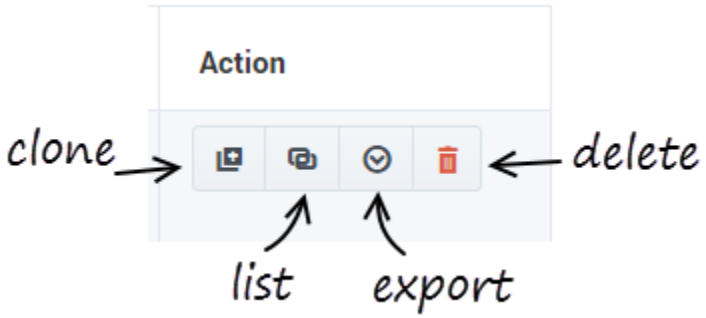
> **Important**
> If you make changes to a media collection, it must be re-published for the changes to take effect in the applications using those resources.

After you click **Publish Media Collection**, Designer opens the **Publish - Media Collection difference** window, where you can use the **Added**, **Deleted**, and **Modified** tabs to review the changes made to media resources since the last time the collection was published.

When you are ready to publish, click **Publish** to publish only those items in the collection that have been changed. Select **Publish Entire Media Collection** if you want to publish all resources contained in the collection.

Managing media collections

Use the **Action** toolbar to manage your media collections:



Clone a media collection

Creates a copy of the media collection while preserving the original. For example, you might want to use a media collection in another application but need to make a few modifications to its resources, such as using an alternate media file or language.

List media collection consumers

Displays a list of the applications that are using the media collection. This lets you see which applications would be impacted if you make any changes to the collection.

Export a media collection

Use this option to download a zip file containing all of the media files in the collection. The zip file includes a CSV file that you can edit to make offline changes to the media collection before importing the collection back into Designer.

Delete a media collection

Click the trash icon to delete a media collection. Designer will ask you to confirm this action, and display a list of applications and modules that are using any of the media resources contained in the collection.

Offline management (CSV file)

You can export a media collection to make changes to it offline. The collection can then be imported back into Designer.

When you export a media collection, a CSV file is included with the exported zip file. You can use an application like Microsoft Excel to edit this file and make changes to the media collection. For example, you can add new resources, mark existing resources for removal, or modify certain resource properties.

There are certain rules to follow when working with the CSV file:

- Don't make any changes to the spreadsheet schema. For example, do not make changes to the column headings or re-arrange the columns.
- When adding a new resource, you can add a new row. However, do not add (or remove) any columns. Also, do not remove rows for existing resources — there is a proper way to remove resources (see Removing resources from a collection).
- All mandatory fields must contain a value before the audio collection can be imported.
- Do not remove or rename any of the extracted files or folders.
- Make sure any *new* media file resources you are adding have been placed in the **media** folder before preparing the zip file and uploading the collection to Designer.
- Items in CSV files are separated (or delimited) by commas. If you need to use a comma within a value, you must enclose it in double-quotes (",").

CSV file fields and descriptions

Field	Description
Mark Media Resource For Deletion	**Mandatory**. Indicates if the media resource should be removed from the collection. By default, all values are set to FALSE. If you change this value to TRUE for a resource, it will be deleted during import.

Field	Description
Audio Resource ID	**DO NOT MODIFY**. This value represents the unique ID generated by Designer for each audio resource. If you are adding a new audio resource, leave this field blank.
Audio Resource Name	**Mandatory**. Name of the audio resource. You can modify this value.
Audio Resource Description	Description of the audio resource. You can modify this value.
Audio Resource Tags	Tags used to group the audio resources for easy identification. Tags should always be enclosed within double-quotes ("tag"). Multiple tags can be separated by a comma, but kept within the double-quotes: "tag1, tag2, tag3".
Mark Language For Deletion	**Mandatory**. Indicates whether the language should be deleted from an audio resource. By default, all values are set to FALSE. If you change this value to TRUE for a language, it will be removed from the associated audio resource during import.
Play Only Text	**Mandatory**. Indicates whether the audio resource should play only text. If set to FALSE, the resource plays as an audio file. If the audio file cannot be played, the text specified in `Text to Be Played` is played.
Audio Resource Language	**Mandatory**. Indicates the language that the audio file/text supports. The language can be any one of the languages supported by Designer.
Audio File Name/Text to Play (if Play only Text is true)	**Mandatory**. If `Play Only Text` is set to TRUE, enter the text to be played. If `Play Only Text` is set to FALSE, enter the name of the audio file to be played. Make sure that the audio file specified here is contained within the **audio** folder.
Text to Be Played	Indicates the text to be played if there is issue playing the audio file. Enter the TTS text to be played.

Adding new resources to a collection

To add a new resource to a collection, add a new row to the CSV file. Specify values for all of the mandatory fields, and any optional fields as desired. Do not enter a value for **Audio Resource ID** as this value is added by Designer after import.

If you are adding an audio file resource, make sure that the file is placed in the **audio** folder before performing the import.

Removing resources from a collection

To remove an audio or language resource from a collection, change the value of the **Mark Audio**

Resource/Language for Deletion field to TRUE.

Supported Audio Formats in Browsers

The types of audio files that you can play is dependent on the web browser that you are using. The following table shows which audio file formats are supported by each browser.

> ### Important
> The table below applies to your browser only and does not indicate whether the file format is supported by GVP. It is important to note that GVP is responsible for playing the audio resource when an application is executed.

Format	Chrome	Firefox	Internet Explorer	Safari
mp3	Yes	Yes	Yes (IE 9 and later)	Yes
wav (16 bit mono)	Yes	Yes	No	Yes
wav (u law)	Yes	No	No	Yes
vox	No	No	No	No

Message Resources

> **Important**
> This item is only available if your site supports digital application types.

Click **Message Resources** in the navigation bar to manage your message resources.

Message Resources are predefined standard responses and user-defined messages that you can use in digital applications and shared modules. A **Message Collection** is a collected set of individual message resources that can be accessed by a digital application (for example, the **Chat Resource Set**).

From this page, you can centrally manage these message resources for all of your applications. If you make any changes to a message resource, the change takes effect the immediately across affected applications.

(See the Applications or Shared Modules page for more information on how to assign a message collection to an application or shared module.)

Creating Message Collections and Message Resources

To create a new message collection, click **Add Message Collection** and enter a name. When you are done, click **Create and Open** to open the new message collection and add message resources.

Next, click **Add Message Resource** and enter a name for the message resource. Make sure to use a unique name as you won't be able to add it if it has the same name as an existing system resource. Click **OK** to save the new message resource.

Properties of the new message resource are displayed to the right of the list. You can choose to add a description to describe the purpose of this message resource. You can also add a tag to relate this message resource to similar message resources with the same tag.

Speech Grammars

Click **Speech Grammars** in the navigation bar to upload and manage speech grammars for use in your applications. Genesys Designer supports **voice** and **dtmf** speech grammars in either **SRGS** (GRXML) or **SLM** format.

A speech grammar defines the list of phrases or options that the caller can input when they use your application. You might use a **voice** speech grammar to tell your application which words or phrases might be used by the caller, to help the application determine how the call should be routed.

For example, an automotive company might upload a speech grammar that contains phrases that a caller might use when they contact the company, such as "I want the parts department" or "I am interested in buying a new car." Your application can use these phrases to determine the best routing target for this customer.

Creating a Speech Grammar

Click **Add Grammar** to create a speech grammar. In the pop-up window, enter a name for the speech grammar and click **OK**.

You can now upload your speech grammar file. In the **Grammar detail** area, click the **Language** drop-down to select the language of the speech grammar. Next, click the **Choose file to upload** button that appears when you hover over the **No file** text. Choose a file to upload.

After the speech grammar file has uploaded, you can click to view the contents of the grammar file in a read-only window.

Next, you can tag the speech grammar or enter a description. You can also specify the format and mode of the speech grammar.

> ### Important
> Genesys Designer does not validate the contents of the speech grammar. The **Grammar Format** and **Grammar Mode** settings only determine how Designer treats the speech grammar in an application.

When you are done, click **Save** to save your changes.

Using Speech Grammars

You can use the User Input block to reference your speech grammars in applications.

In the **Input** tab, you can select **External Grammars** and click **Add Grammar** to add your speech grammar to the block. You can use multiple speech grammars at once.

In the **ASR Settings** tab, you can enable the **Use application-wide ASR settings** check box to use the default ASR (Automatic Speech Recognition) settings. You can define these settings by clicking **Settings** in the Toolbar.

Alternatively, you can disable the **Use application-wide ASR settings** check box to fine-tune the ASR settings for this **User Input** block.

Refer to the User Input block page for more information.

Business Controls

The **Business Controls** pages enable you to manage resources and settings that are specific to your site or business, such as Emergency Flags, Business Hours, Special Days, and Data Tables. Any changes made on these pages are immediately applied to your applications and modules—you do not have to update each application or module.

Learn more about:

Emergency Flags

Check for and react to an emergency condition.

Business Hours

Specify when your business is open or closed.

Special Days

Create holidays and other special days.

Data Tables

Create and manage data tables.

Callback Settings

Business Controls is also where you can manage the settings for Callback V2 (if applicable to your environment).

When you add or modify the Special Days or Business Hours for a callback virtual queue, Designer stores the values for each virtual queue in a system-defined data table called CALLBACK_SETTINGS. You can then manage the settings for each callback virtual queue by editing its related row in the table.

For more information about Callback, see the Callback V2 blocks page.

> ### Tip
> The CALLBACK_SETTINGS data table contains default values for parameters that are not set in **Business Controls** or by the Callback V2 blocks.

Emergency Flags

The Emergency Flags feature lets your applications check for and react to an emergency condition.

For example, if your business might be closed due to a storm, you can create a common Emergency Flag in Business Controls and have one or more of your applications check it. If the Emergency Flag is set, you can add special handling for that condition.

When the Emergency Flag is activated, the applications receive the new status when they check it, and start handling the emergency accordingly.

Adding an Emergency Flag

Click **Add Emergency Flag** to create an Emergency Flag. Enter a name in the provided field and click **OK**.

Activating or Deactivating an Emergency Flag

To activate or deactivate the Emergency Flag, click the On/Off slider.

Checking for Emergency Flags

To check the Emergency Flag in an application, add an Emergency block to the **Application Flow**.

In the properties of the **Emergency** block, enable the **Use Emergency Flags defined in Business Controls** check box and select the Emergency Flag that you previously created. The block will then show the current status of the Emergency Flag (this status display is for informational purposes only; you can't modify it from within the block).

The next time you activate this Emergency Flag from the **Business Controls** menu, the Emergency Flag handling is also activated in your application.

> **Important**
> Activating an emergency flag can impact existing calls if the check was done prior to activation.

Business Hours

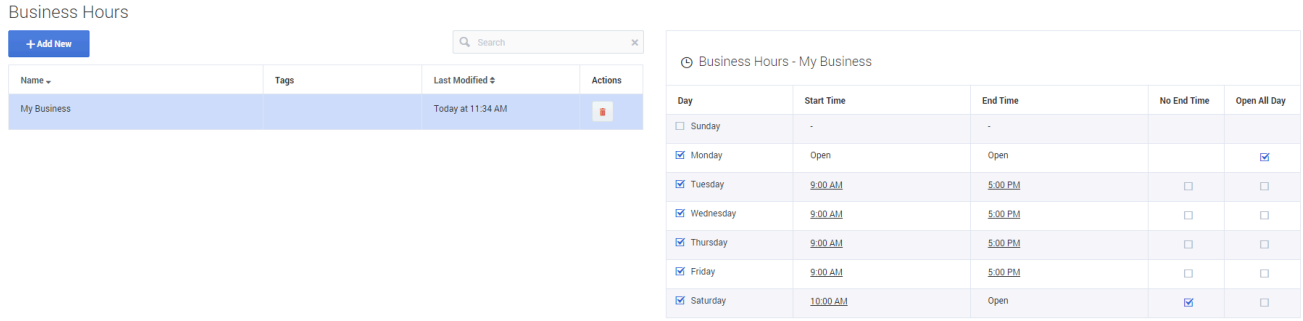
You can use the Business Hours feature to create various sets of business hours for use in your applications.

You must use a unique name for each set of business hours. You might want to use a combination of company names and departments. For example, you could use `CompanySales` and `CompanyService`.

It is also recommended to use tags to help organize your business hours.

Specify open and closed times

Use the check boxes to indicate which days your business is open. Click the **Start Time** and **End Time** values to specify the opening and closing times, or use the **No End Time** or **Open All Day** options.



Specify alternate business hours

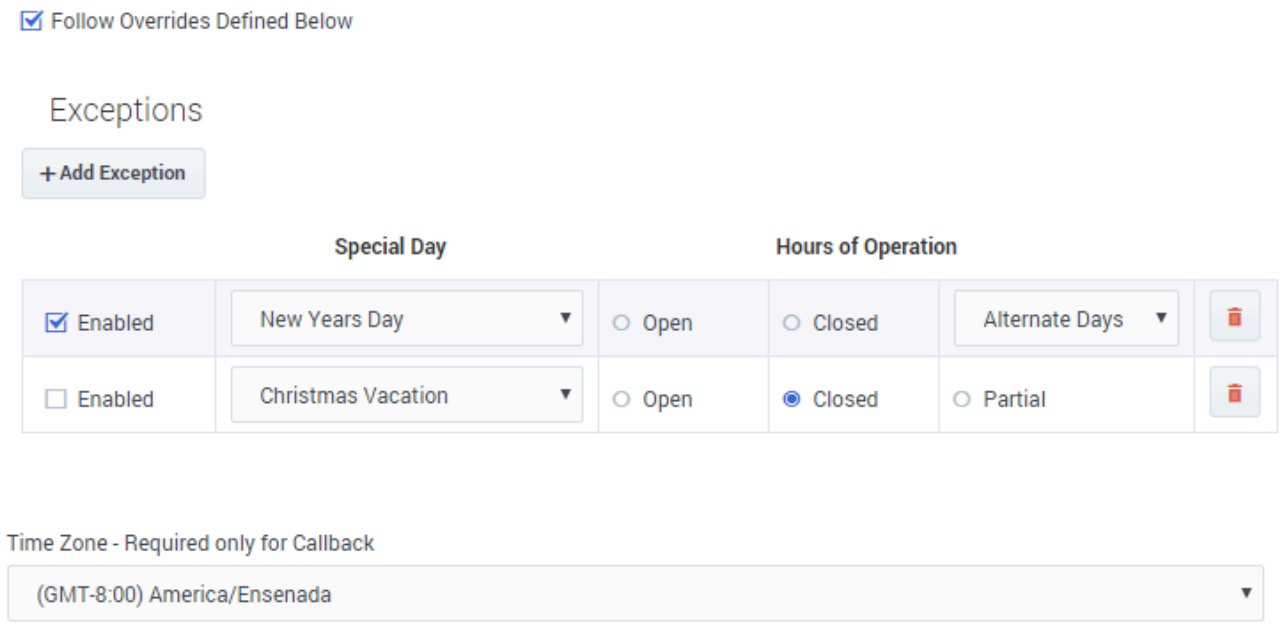
You might want to specify business hours that differ from the normal Special Day hours. To specify alternate business hours for Special Days, select **Follow Overrides Defined Below** and add an override.

Select a Special Day and choose the **Hours of Operation**.

> ### Warning
> You must select a Special Day for each exception. Otherwise, the exceptions will not take effect.

Exceptions (or overrides) are enabled by default, but you can uncheck the **Enabled** box to disable them.

If you are using Callback V2, you can select an alternate **Time Zone**. (This setting is only used by Callback V2, to provide an exception to the current Designer time zone setting. It doesn't change the time zone settings for any other feature or application.)



Once you have set Business Hours, you can use them in your applications by using the Business Hours block.

Special Days

You can use **Special Days** to create holidays and other special days for use in your applications.

Add a Special Day

Click **Add Holiday** to add a special day. Use a name that describes the special day, such as `New Years` or `Thanksgiving Day`, and then use **Add Date Range** to create and specify a date range. You can create and define multiple date ranges for a Special Day, as well as enable or disable them.

As you make your changes, the **Special Day** is automatically saved. If the special day occurs on the same day every year, select the **Occurs every year** check box.

If you make an error, or want to make changes, you can edit or delete the Special Day by selecting it.

Once you have set Special Days, you can use them in your applications by using the Special Days block.

Data Tables

A Data Table is similar to a spreadsheet. It is a two-dimensional array that is filled with values that can be read by a Designer application. Each data table has at least one key (primary key) column. The value of the key column is used to look up a row in the table.

Data Tables are useful if you want an application to refer to values that are stored outside of the application, or if you want an application to update values in a table without changing values in the application.

Clicking a Data Table opens it for editing. Hovering on it will let you choose whether to open it in **Edit** or view **Read-only** mode.

> **Tip**
>
> - If another user has a Data Table open for editing (locked), you can still open it for viewing in **Read-only** mode.
> - If you prefer to work on your Data Table in another program, such as Microsoft Excel, you can export the Data Table into a CSV file. See Exporting and Importing Data Tables.
> - You can create and modify Special Days and Business Hours directly from within a Data Table. The changes take effect when you save the table.

Recommendations and guidelines

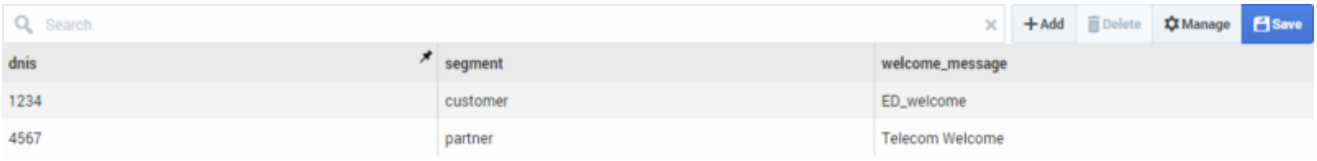
Here are some important things to keep in mind when working with data tables:

- Some browsers might display a script error or temporarily freeze when opening large Data Tables. This is normal, and you can let the script continue or wait for the browser to finish loading the data table.
- Limit the number of rows to 1000 and the total size of the data table to no more than 10,000 cells. If the number of rows is less than 1000, you can increase the number of columns until the 10,000 cell limit is reached. For example, a 200 row table can have up to 50 columns, and a 1000 row table can have a maximum of 10 columns.
- There are limits to how much data can be stored in a data table (a data table is not meant to be used as a full-scale database). Therefore, try to focus more on data that needs to be changed or updated frequently, or are critical to your business.
- Clearly categorize the data that you want to store. For example, if you are storing customer profiles, some various categories could be **Name**, **Address**, and **Phone**. Then you could set up **Name** and **Address** as a *string* data type and **Phone** as a *numeric* data type.
- Clearly define the lookup keys, as these are important for searching for (and locating) the relevant data.

- Note that you can't use a value of "0" (zero) in a numeric or integer key column (this returns a validation error).
- Take advantage of Designer's ability to let you carefully review and verify your changes before committing them to the data table.
- After a data table is published, you cannot change the data types of the existing columns. You can, however, still modify the schema of the data table and change the data types of columns that have not yet been published.

Adding a Data Table

To add a new Data Table, click **Add Data Table** and enter a name for the Data Table. For example, you can create a Data Table to segment customers based on a DNIS key.



After you click **OK**, Designer creates the Data Table and it appears in the Data Tables list.

Next, click your Data Table to open it. Designer prompts you to define a scheme for your new Data Table (for example, the names of columns for your table).

Click **Manage Schema** and configure the following options:

- **Column** - The name of a column to add to your Data Table. For this example, enter `dnis`.
- **Display Name** - Lets you customize how the column name is to be shown in the Data Table (this does not overwrite the actual **Column** value). For this example, enter `Dialed Number Identification Service`.
- **Key?** - If enabled, this column is a key column and is used to look up a row of values. For this example, enable this check box.
- **Data Type** - Specifies the type of value(s) that will be used by this column. Supported data types include **string**, **numeric**, **boolean**, **announcement**, **integer**, **datetime**, **datetimerange**, **skillexpression**, **timezone**, **businesshours**, and **specialdays**. For this example, select **numeric**. (When specifying integer values, the **numeric** data type does support integers, but data table lookups complete faster if you use the **integer** data type for these values.)
- **Description** - An optional description of the column.

Click **Add a column** and add two more columns: **segment** (type is string) and **welcome_message** (type is announcement). Do not enable the **Key?** check box.

When done, click **OK**. You have created a Data Table with a key column of **dnis** and value columns of **segment** and **welcome_message**. Click **Add** to add a row in your Data Table. You can now click the cells under each column to enter values.

Once you have created a Data Table, you can place a Data Table block in an application.

Editing Data Tables

You can change the value of a table cell by clicking on it. As soon as you start editing a cell, the row is automatically selected and the updated text is in blue.

If you expand an updated row, the original values (crossed-out) are shown below the modified cells. This lets you compare the new value to the one that was previously saved.

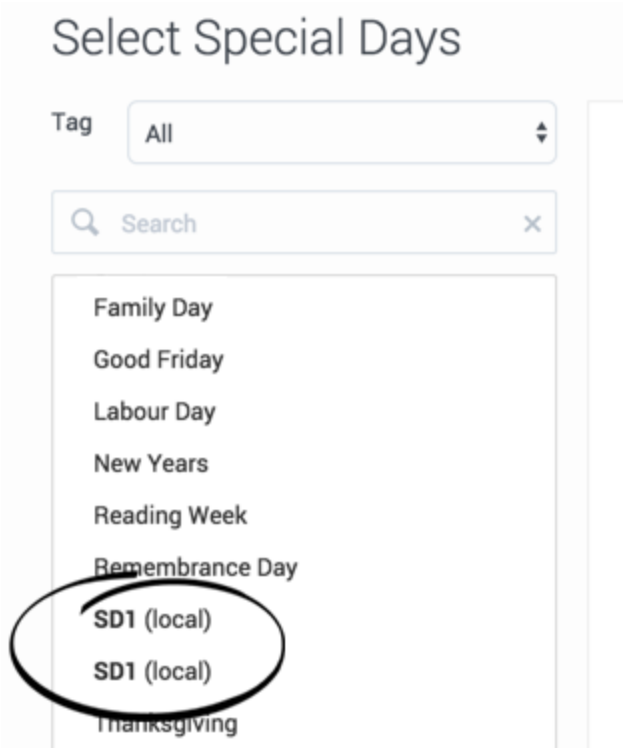
Tip

Looking for something specific in the Data Table?

- Each column has a search box you can use to look for a specific item. As soon as you start typing, Designer hides all other rows except for the ones that contain matches to what you have entered.
- At the bottom of the page, there is a **Row Count** box you can use to jump directly to a specific row number.

You can also create and modify Special Days and Business Hours directly from within a Data Table. The changes take effect when you save the table.

Note that if you add new Business Hours or Special Days to a data table (for example, you click a cell under a Special Days column and the Select Special Days picker opens), the new business object is *local* only to that data table — that is, it won't show up in the *global* Special Days list and be available to other data tables. Local business objects are in **bold** and have (local) beside them.



Adding and Removing Rows

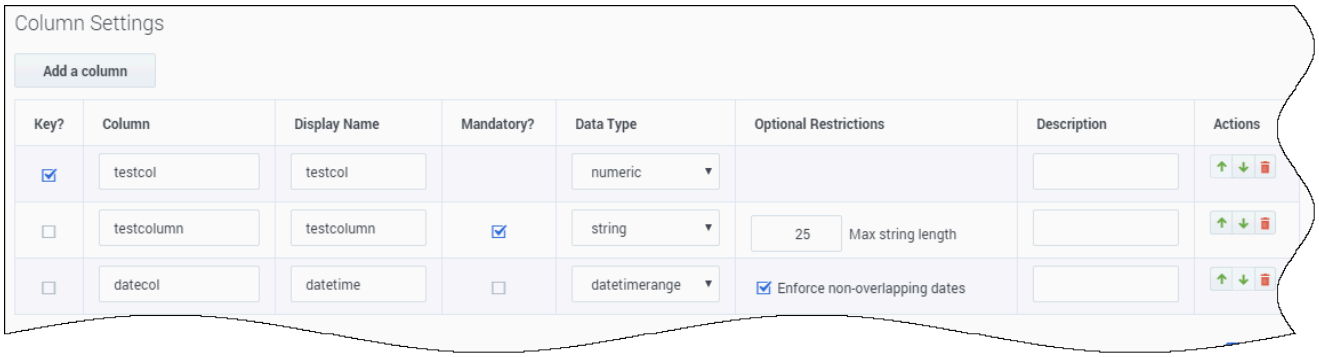
To add a row, click **Add Row**. To remove a row, select it and click **Mark For Deletion**. Any rows that you mark for deletion are removed the next time you save the table.

Changing Column Settings

To make changes to the columns, click **Column Settings**. You can add new columns, or update the properties of existing columns. For example, you can update the **Display Name** of a column, indicate if it is mandatory, or specify any **Optional Restrictions** for that particular data type, such as a maximum string length (for **string**) or whether to enforce non-overlapping dates (for **datetimerange**).

Use the options under **Actions** to change a column's position in the grid or delete it.

> ### Important
> If you change the data type of a column, make sure that after saving the data table, you refresh or reload the page before entering or editing any cell values. Otherwise, the cell values under the modified columns might not display correctly. **After a data table is published, you cannot change the data types of the existing columns.** You can, however, still modify the schema of the data table and change the data types of columns that have not yet been published.



Reviewing Your Changes

At the bottom of the data table, a color-coded counter keeps track of the number of rows you have added, modified, or marked for deletion.

To view only the rows that were added, changed, or marked for deletion, check the **Show Modified Rows Only** box. Uncheck it to go back to editing mode.

You can also review changes to local business objects by clicking **Display Business Object Diffs** and selecting **Business Hours** or **Special Days**. You can then select an item from the list to view the original version side-by-side with the revised version. On the original version, edited properties are highlighted in red to indicate edits and deletions. On the revised version, edited properties are highlighted in green to indicate edits and additions.

Saving and Publishing

When you are ready to commit your changes, click **Save Table**.

> ### Important
> Make sure to review your changes! After you click **Save Table**, the changes can't be undone.

When you save the data table, Designer validates your changes and lets you know if there are any errors. Canceling discards all unsaved changes and restores the table back to its previously-saved state.

> ### Tip
> When saving a data table, you might see some of the values (particularly for Business Hours or Special Days) suddenly change to "N/A". This is just temporary, and the correct values will appear after the save operation completes.

Saving a data table only preserves the changes you have made. To activate the changes in the live production environment, click **Publish**. After you publish the data table, the applications that reference it have access to the latest changes.

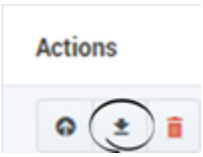
Exporting and Importing Data Tables

You might prefer to use another program, such as Microsoft Excel, to edit Data Table values. If so, you can export a Data Table from Designer into a CSV file that can be edited in Excel. When you are done, you can import the edited CSV file into Designer.

> ### Warning
> When importing the edited CSV file, do not change the schema (structure) of the data table in Designer. The data table schema must remain unchanged between the export and the subsequent import. Otherwise, the import will fail.

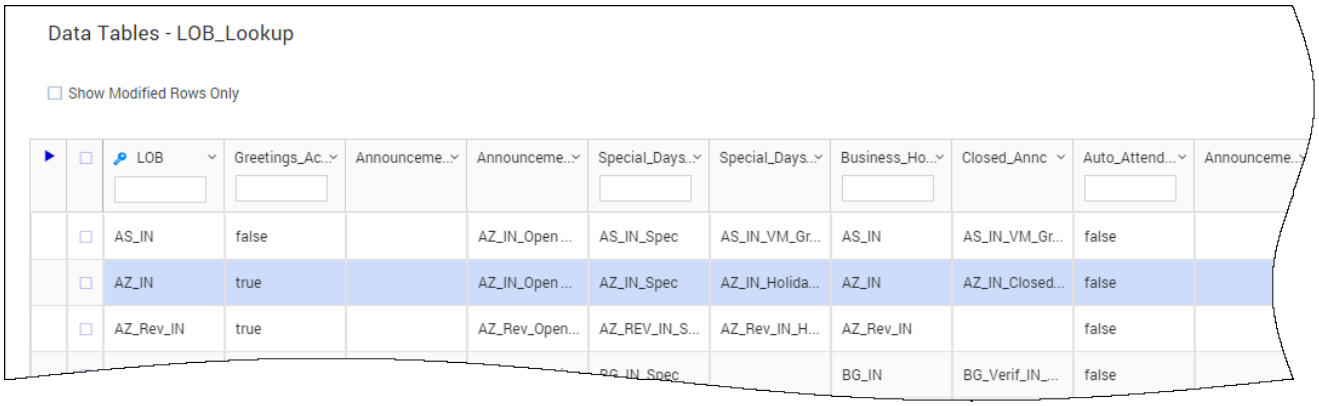
Export

Click **Export** in the **Actions** column to export a Data Table from Designer into a CSV file.



Below is a sample Data Table, its generated CSV file, and the CSV file in Microsoft Excel.

Data Table



CSV File

Here is how the row that is highlighted above would appear in the exported CSV file:

```
AZ_IN,true,,dbc63d70-37d6-11e6-a888-e53edc8cf09b,AZ_IN_Spec,2165b9f0-37d7-11e6-a88
8-e53edc8cf09b,AZ_IN,0508d4e0-37d7-11e6-a888-e53edc8cf09b,false,,,,VQ_AZ_IN,AZ_IN,
30,AZ_IN,30,,,true,EstimatedWaitTime,78e070d0-37d7-11e6-a888-e53edc8cf09b,a38864a0
-37d7-11e6-a888-e53edc8cf09b,,true,false,,Arizona_IN_transaction,e5ffb100-37d5-11e
6-a888-e53edc8cf09b
```

Note that some of the items are represented by their resource ID and not their actual name. For example, the audio resource *AZ_IN_Open Greeting* appears as "dbc63d70-37d6-11e6-a888-e53edc8cf09b". This ensures that the correct resource is being referenced (names of resources can be changed, but their assigned resource IDs always remain the same).

> **Tip**
>
> Items in CSV files are separated (or delimited) by commas. If you need to use a comma within a value (such as for the text in a script dialog) you must enclose it in double-quotes (","").

Data Table in Microsoft Excel

Here is how the CSV file appears when viewed in a program like Microsoft Excel:

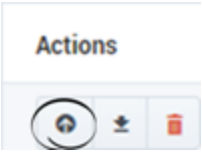
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	LOB	Greetings	Announce	Announce	Special_D	Special_D	Business_	Closed_Ar	Auto_Atte	Announce	Priority_S	Priority_Ir	Target_Vi	Target_1
2	AS_IN	FALSE		dbc63d70-	AS_IN_Sp	bc6259f0-	AS_IN	bc6259f0-	FALSE				VQ_AS_IN	AS_IN
3	AZ_IN	TRUE		dbc63d70-	AZ_IN_Sp	2165b9f0-	AZ_IN	0508d4e0-	FALSE				VQ_AZ_IN	AZ_IN
4	AZ_Rev_I	TRUE		006bd7b0-	AZ_REV_I	44e28ba0-	AZ_Rev_IN		FALSE				VQ_AZ_Re	AZ_REV_I
5	BG_IN	TRUE		2774bfe0-	BG_IN_Spec		BG_IN	90a14ca0-	FALSE				VQ_BG_IN	BG_IN

After you have edited the CSV file, you can import it into Designer.

> **Tip**
>
> While you can edit any item listed in the CSV file, it is more practical to edit items referenced by resource IDs from within the actual data table.

Import

Click **Import** in the **Actions** column to import a CSV file into a data table.



Important

- Import is disabled for data tables that contain **menu** data types. If you do not see the **Import** icon in the **Actions** column for a data table, it indicates that the data table is using the **menu** data type.
- If you are importing a CSV file into an empty data table, Designer designates the first column as the key column.
- If you are importing a CSV file into a populated data table, you must ensure the CSV file and the data table use the same table headers. If the headers do not match, Designer displays an error.

Admin

> **Important**
>
> The **Admin** settings are only available to users who are assigned to the **Administrator** role in Designer. For more information about roles in Designer, see Permissions and Access.

Click **Admin** in the navigation bar to access the Designer Administrator settings.

Partitions

Use partition-based access control (PBAC) to manage partitions, resources, and users.

Partitions

As a Designer Administrator, you can use the **Partitions** page to control the resources that users have access to through **Partition-Based Access Control (PBAC)**.

With PBAC, you can create a partition and assign certain resources to it. In Designer, "resources" are the various objects used during interaction sessions, such as Applications, Shared Modules, Business Hours, Special Days, Emergency Flags, Data Tables, Speech Grammars, and Audio and Message Resources.

For each partition, you can then select the users who will belong to it. The users will only be able to see and manage those resources that are assigned to the partitions they belong to. All other permissions they have (as defined by their assigned roles within Designer) remain in effect.

Each user's PBAC details are stored in their Workspace settings and retrieved by Designer during login.

Watch this short video to learn how PBAC works:

Link to video

Watch this short video to see an example of how PBAC can be set up:

Link to video

> ### Important
> By default, PBAC works by inclusion. If a user is not assigned any partitions, it is assumed that PBAC is not in effect for that user and they will have access to ALL resources, including those that have partitions assigned to them. Similarly, if a resource is not assigned any partitions, it is considered a public resource that is accessible to ALL users.

In general, partitioning can be set up as follows:

- Define a private partition. Assign it to all resources that you intend to control using PBAC. You can leave out any resources that should remain globally visible.
- **Don’t assign this partition to any users.** This private partition will ensure that resources under partitioning control will NOT be visible to a user who has at least one partition defined.
- For each department, set up a dedicated partition and assign it to users from that department. Then assign each partition to the resources those users need access to. (Here's an example.)
- New resources inherit the partitions of the users who created them, and remain accessible only to users who belong to that partition.

For example...

You might create a partition for each of the following departments:

- Sales
- Finance
- Marketing

Then add users as members to their appropriate partitions:

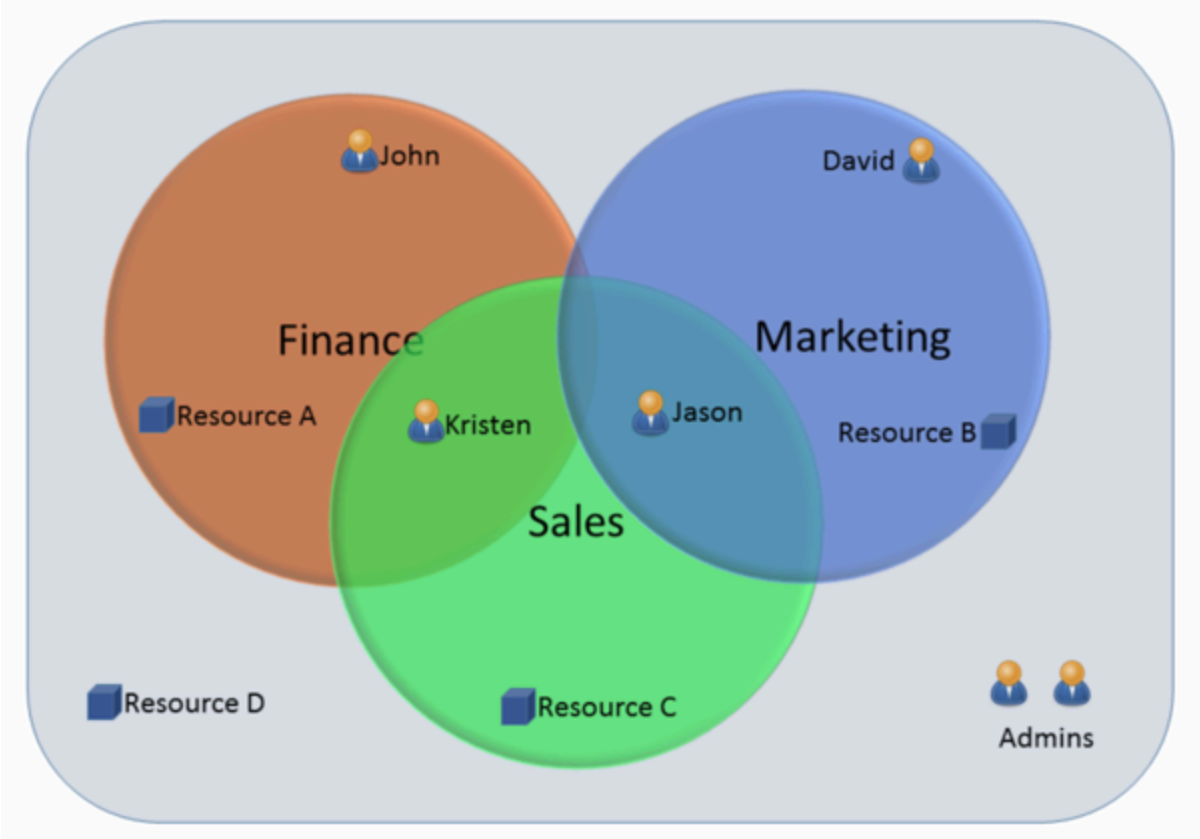
- John to Finance
- David to Marketing
- Kristen to Sales and Finance
- Jason to Sales and Marketing

Remember that users who are Designer Administrators do not need to be assigned to a partition as they already have full access.

You can then assign certain resources to each partition:

- Resource A to Finance
- Resource B to Marketing
- Resource C to Sales
- Resource D to "none" (remember that non-assigned resources are visible to ALL users)

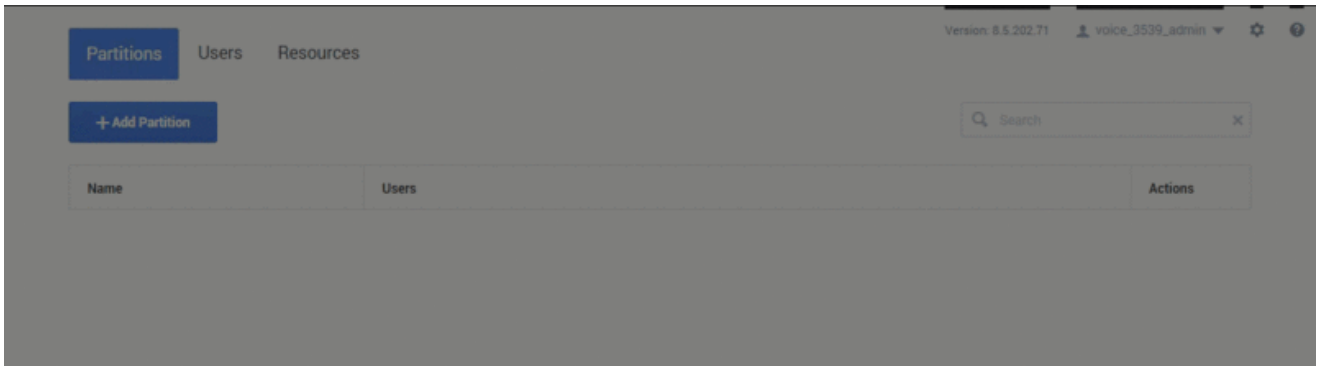
This chart illustrates the relationships between the users, resources, and partitions described in this example:



Partitions tab

Use this tab to add or manage partitions and select the users who can access them.

For example, to add a new partition called **Sales**:



After the partition is added, you can use the **edit users** action to select the users who can access it:

> **Tip**
> Users who are also Designer Administrators don't need to be assigned to partitions as they already have full access. Even if they are assigned to partitions, they will continue to see all resources as if they were not.

Des admin partitions addusers.gif

Users tab

Use this tab to view the list of users and manage their assigned partitions.

For example, to assign **user_sales** to the **Sales** partition and remove them from **Finance**:

Des admin users edit.gif

Resources tab

Use this tab to view the list of resource types and their associated partitions.

For example, let's say the Business Hours resource **regularhours** is already associated with the **Service** and **Sales** partitions, but now we want to also associate it with **Marketing**:

Des admin resources edit.gif

> **Important**
> There are certain Designer resources that cannot be assigned to a partition because they are used by the system or are common resources that are shared across multiple applications. These include templates, shared audio resources, and system-based data tables (such as CALLBACK_SETTINGS and

> NUMBER_VALIDATION_CONFIGURATIONS). All users have access to these resources.

Using Blocks

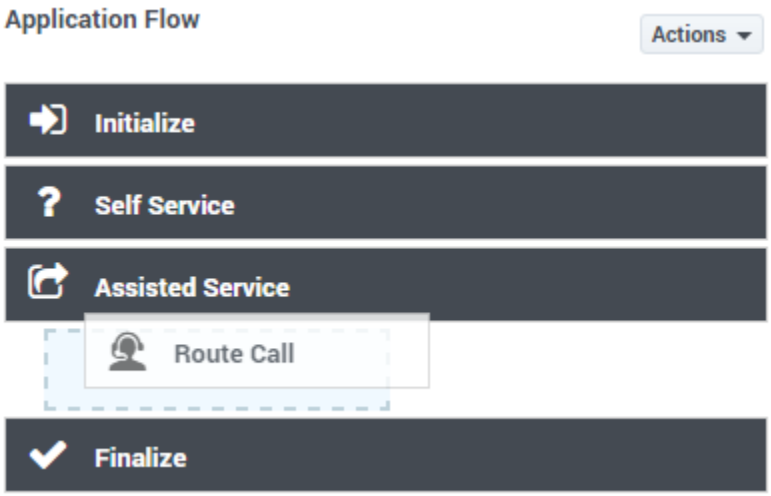
This chapter provides information about the blocks that are available in Designer.

> **Important**
> Some blocks might not be available if you are creating an **IVR** type application. Refer to the Applications page for a list of blocks that cannot be used in **IVR** type applications.

Build Logic

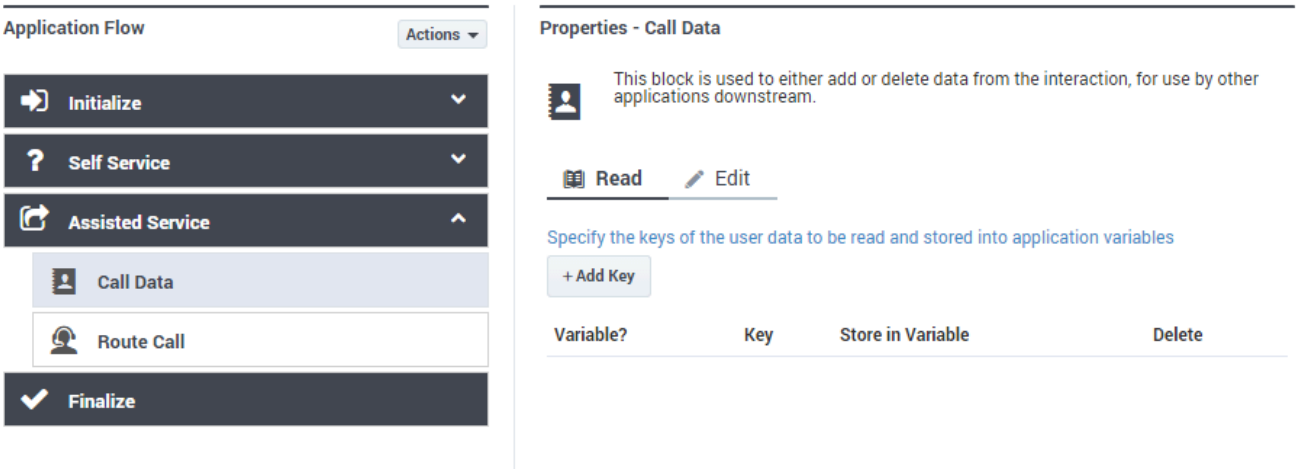
Using the Palette

The blocks are grouped into various categories on the **Palette**. You can drag any block from the **Palette** into the **Application Flow** and place it under the phase in which you want it to execute. If the block can be used in that phase, a blue placeholder block appears and you can drop the block to place it in that phase.

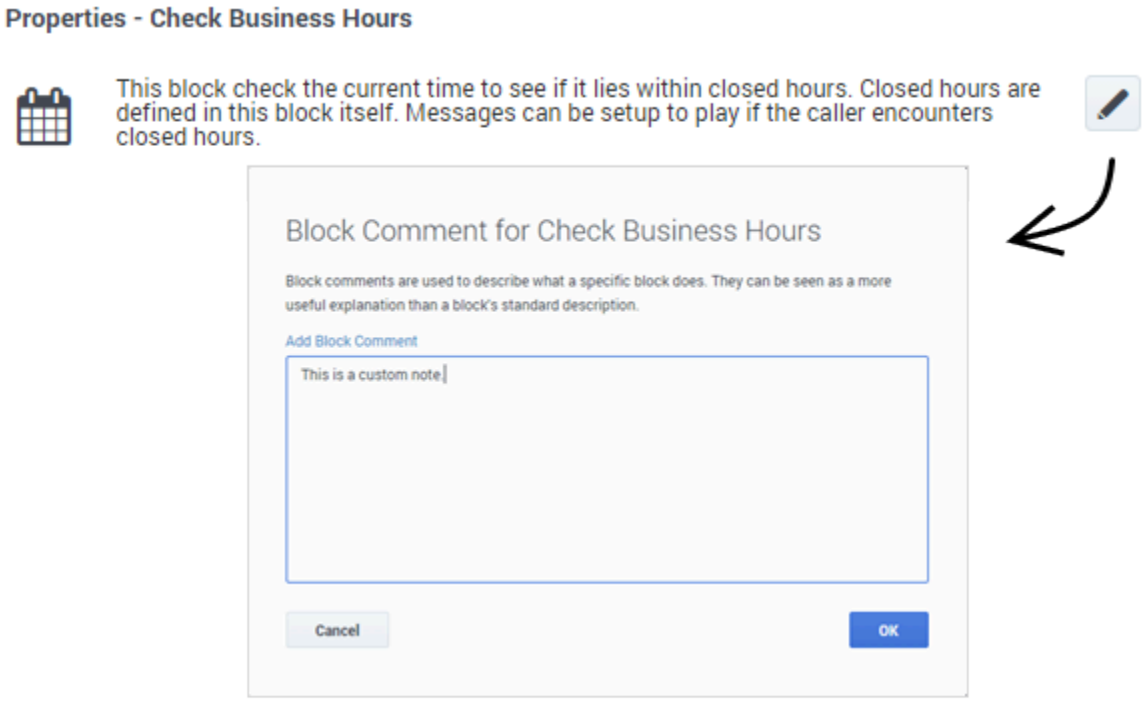


After placing a block, its details are shown in the **Property** view and you can configure the block and provide your application logic.

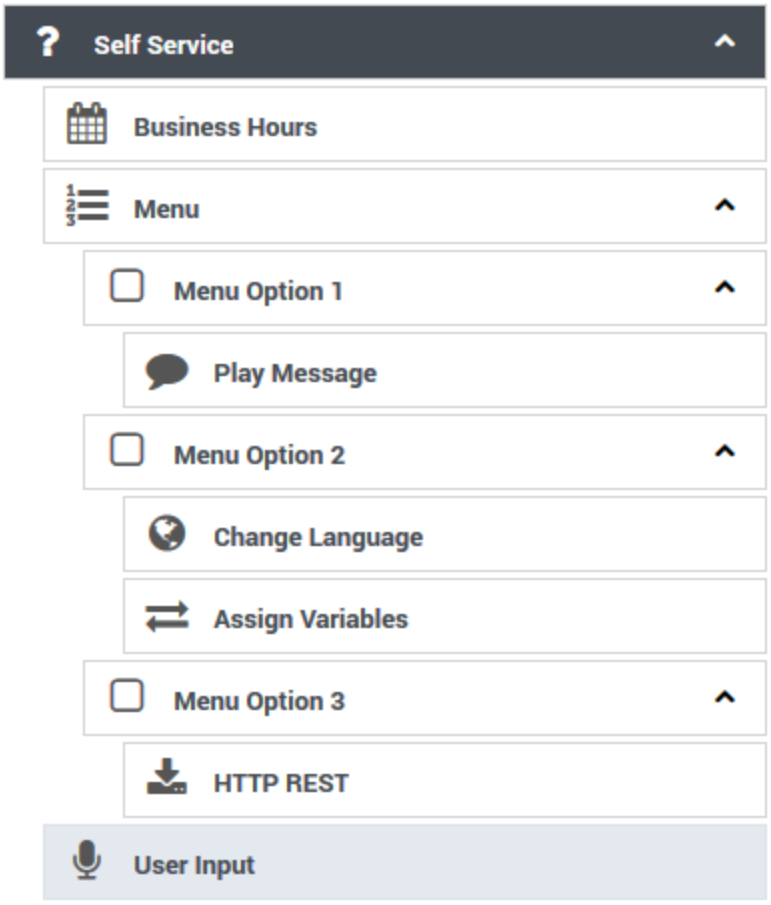
Click a block in the **Application Flow** at any time to select it, highlight it, and show its details.



Each block has a default description, which you can edit to add your own description or comment.



You can place child blocks underneath some blocks. Child blocks are indented underneath their parent block. With the **Menu** and **Segmentation** blocks, this indicates that several outcomes are possible but only one path is followed when the application runs.



In the above image:

1. A user hears a menu prompt.
2. The user enters input corresponding to one of the available menu options.
3. The child blocks of that menu option execute before the application continues with the next block after the **Menu** block.

In this case, if the user chooses **Menu Option 2**, the **Change Language** block runs, followed by **Assign Variables**, and then the menu completes and moves onto **User Input**. The child blocks under **Menu Option 1** and **Menu Option 3** do not execute in this scenario.

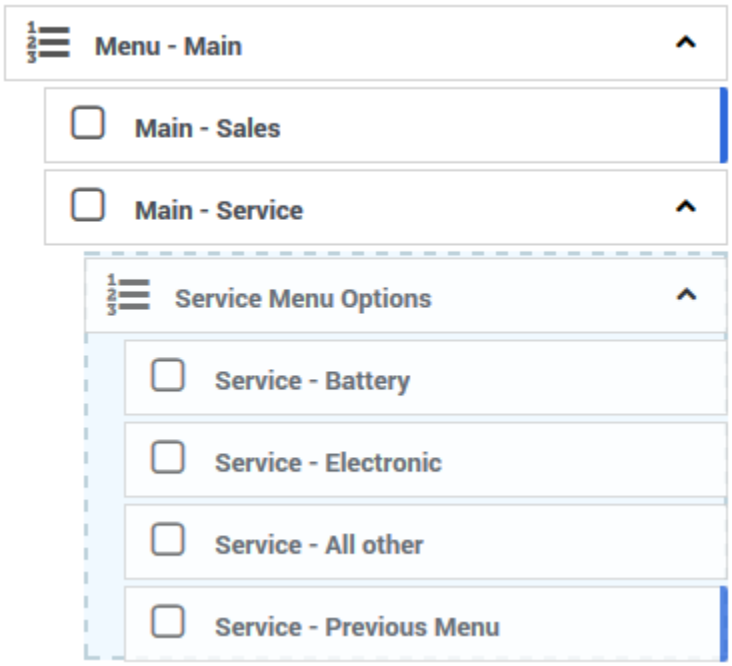
In general, an indentation on the **Application Flow** canvas indicates that a decision or branch occurs, and one of several mutually exclusive paths is followed.

It is also possible for **Menu** or **Segmentation** blocks to be nested within one another - that is, a top-level menu option can lead to a second menu. This is indicated with multiple levels of indentation on the **Application Flow**. Once an option or a branch completes, the application returns one level higher and continues execution.

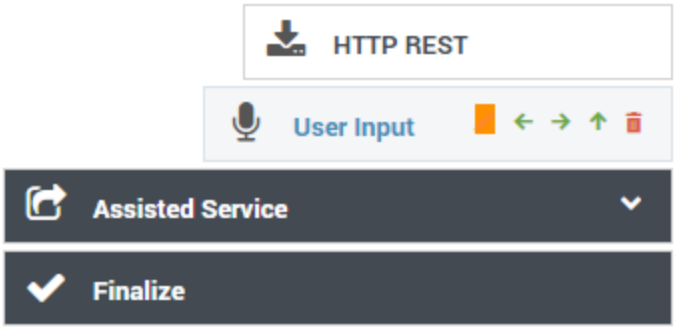
> **Tip**
> When you save your application, Designer also remembers if a group of nested blocks was expanded or collapsed. So the next time the application is opened for editing, the blocks appear in the same state as they were during the last save.

Moving and arranging blocks

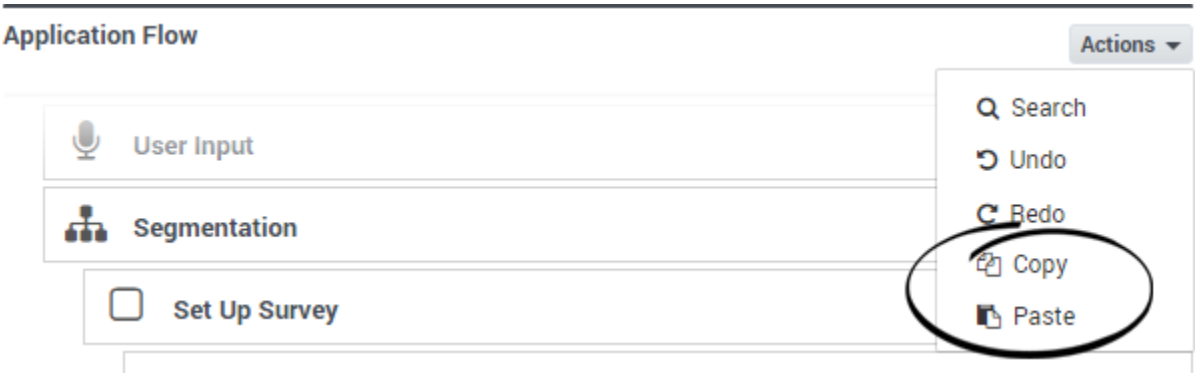
To rearrange the order of blocks, you can drag and drop blocks around the **Application Flow**. Moving a parent block also moves any child blocks under it, so you can move entire groups of blocks together in one operation.



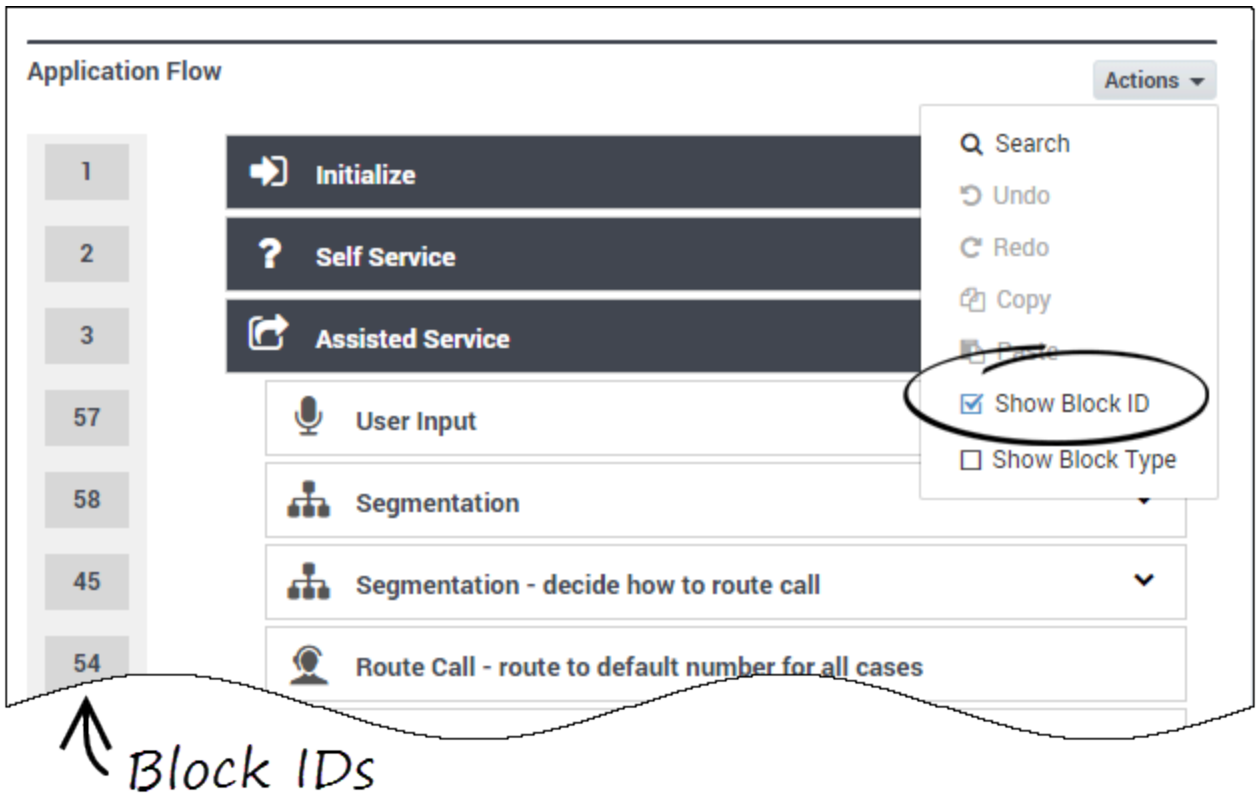
You can also move groups of blocks by clicking the arrows that appear when you hover over a block. This is useful if an application becomes large enough that dragging and dropping is unfeasible. These arrows show allowable operations on a block: up and down to move a block backward and forward within a phase, and left and right to change the indentation of a block underneath a parent block.



You can also use the **Copy** and **Paste** functions (under **Actions**) to copy a block to another location in the flow, or to another module or application. Copying a parent block also copies any child blocks under it, so you can copy entire groups of blocks together in one operation. Keep in mind that blocks can only be copied to locations where that type of block is permitted.



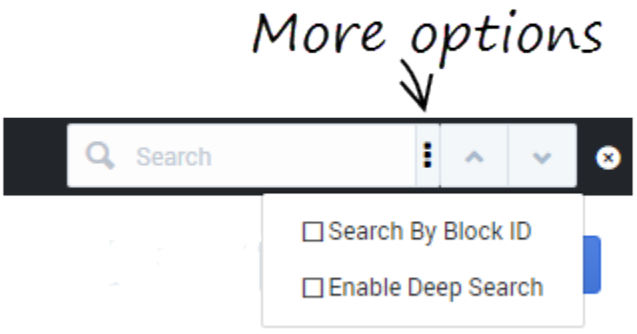
Under **Actions**, you can select **Show Block ID** or **Show Block Type** to toggle the visibility of those attributes. For example:



Searching

To search the blocks in the application flow, select **Search** from the **Actions** menu. The search box appears in the main navigation bar and you can start typing the search term you are looking for.

If you click the more options button, you can also select the option to search for blocks by their numeric Block ID.



The results are highlighted in the application flow, and you can use the up/down buttons to jump to the next or previous result.

Limiting Application Indentation

Although Designer allows you to use several levels of indentation, you might not be able to access the block properties element if you have more than 10 levels of indentation.

To prevent your application from becoming too deeply indented, use Menu and Segmentation blocks to jump to specific points in the application. This takes control back to the main *trunk* and prevents the application from being too indented and difficult to understand. **Menu** and **Segmentation** blocks that do not terminate the application within a reasonable depth should include a Go To block to jump to a different part of the application.

> **Important**
>
> In certain cases, you might need to skip over certain parts of the main application. In these cases, use a **Go To** block to forward processing to the correct block in the application.

ECMAScript Expressions

Some block properties accept ECMAScript expressions that are executed by the application at runtime. This allows the application to perform dynamic operations, such as calling an ECMAScript function or combining the values of other variables.

In general, block properties do not support ECMAScript expressions unless otherwise stated. If ECMAScript expressions are not supported, you must enter a value (a string that is taken as a literal). This value is not evaluated at runtime. For example, in the Play Message block, the value of a TTS prompt is taken as a literal string.

Supported Properties

The following lists blocks properties that support ECMAScript expressions:

- Initialize phase - User Variables
- Activity block - Values in key-value pairs
- Assign block - Assignments
- Data Table block - Lookup Key
- Menu Option block - Set Variables
- Milestone block - Values in key-value pairs
- Return block - Assigning values to output variables
- Shared Module block - Assigning values to input variables

Tips

- See ECMAScript Documentation for more information on using ECMAScript expressions.
- Strings must be quoted. For example, `'hello'` is a string, whereas `hello` is a reference to a variable called **hello**.
- Single quotes (`'`) are recommended, as opposed to double quotes (`"`).
- When specifying an object in JSON notation, surround the JSON with parentheses. For example: `({'abc': 'def'})`.

Examples

Below are examples of how you might use an ECMAScript expression in a Designer application.

Building a Dynamic TTS Prompt

You can use the Assign block to concatenate a string to be spoken by the application. The expression below reads the caller's phone number or ID.

```
'You are calling from ' + ANI
```

Control the Application Flow

A Segmentation block can take ECMAScript expressions that evaluate to a Boolean value, and thus control the flow of the application. For example, you might want to inform your customers about upcoming seasonal events and you need a way to determine the current season and whether the event is occurring within the coming week. The expression below determines whether the call was received within seven days of the event, and whether the current season is summer or autumn.

```
numDays > 7 && (isSummer || isAutumn)
```

ECMAScript User Functions

Designer also has built-in ECMAScripts that you can invoke from a Designer application, such as from an Assign or Segmentation block, to perform certain functions at runtime.

isDataTableValueValid

You can use this function to determine if a value returned from a data table query is valid. For example, you might use the following function in an Assign block:

```
isDataTableValueValid(value, datatype)
```

This function has two arguments:

- *value* is a single value returned from a data table query
- *datatype* is the data type of the data table column, such as 'string', 'boolean', 'integer', 'announcement', or 'numeric' (this argument is optional)

If the data table value is valid, the script returns `true`. Here is a list of values that this function can return:

- `isDataTableValueValid(varStr, 'string')` on a valid (or empty) string returns `true`. Anything else returns `false`.
- `isDataTableValueValid(varNum, 'numeric')` on a valid number or `0` returns `true`. Anything else returns `false`.
- `isDataTableValueValid(varNum, 'integer')` on a valid integer or `0` returns `true`. Anything else returns `false`.
- `isDataTableValueValid(varBool, 'boolean')` on `true` or `false` returns `true`. Anything else returns `false`.
- `isDataTableValueValid(varAudio, 'announcement')` on a valid (or null) announcement returns `true`. Anything else returns `false`.

Busy Treatments

A busy treatment is a special form of voice call handling that tells Designer what to do while a caller is waiting for their call to be connected with an agent. For example, you can play music for callers or provide them with updates about their estimated wait times.

Certain blocks allow you to specify audio files or self-service type shared modules as busy treatments.

Route Call and Route Agent blocks

The Route Call and Route Agent blocks both have a **Treatments** tab where you can specify an audio file or a shared module as a busy treatment.

If you choose to add an audio-based treatment, a Play Message block is automatically nested below the routing block. Use this block to select and configure the audio options.

> **Important**
>
> If multiple consecutive **Play Message** blocks are added beneath a routing block as treatments, Designer considers them as one single treatment.

If you choose to add a module-based treatment, a Shared Module block is automatically nested below the routing block. Use this block to select the shared module that will be used as a busy treatment.

Busy treatments defined in routing blocks will loop automatically until a certain condition is met – such as the call is routed, the caller hangs up, or the timeout specified in the routing block expires – at which point the next block in the application is triggered.

> **Important**
>
> After a busy treatment has been executed at least 10 times, Designer exits the routing block and moves to the next block if the average duration of the treatment is less than 1000 ms (for example, due to a missing audio file).

Start Treatment block

The Start Treatment block also lets you specify a busy treatment, but it works a bit differently than the treatments used in the routing blocks.

Typically, you would use this block in the Assisted Service phase when you want to start a busy treatment — for example, play an audio file to callers while they wait to speak with an agent — and then move on to the routing blocks, all without interrupting the playback to the caller.

Things to keep in mind when using this option:

- **Don't define any additional treatments in the routing blocks that directly follow the Start Treatment block.**
  You want the audio started by the **Start Treatment** block to continue playing while the routing blocks do their job. If a routing block starts another treatment, the treatment that is playing stops.

- **The Start Treatment block does not loop a module automatically.**
  If you want to set up looping, you might need to use a GoTo block in the module or find another way to loop it back. For example, you might add additional logic in the module to detect which mode it is being used in. This will expose an input parameter that controls whether the module loops internally or not. Or, you could clone the module and have a different looping logic defined in the cloned module. (However, take note that this option creates copies of the same logical module and might make maintenance more difficult.)

Validation

Designer enforces a drag-and-drop policy to ensure that you can only place blocks into applicable phases. In rare scenarios, a block might be placed in an invalid phase. In these cases, the validation process that occurs after you click **Publish** will report this failure with an error that includes the blocks placed into invalid phases.

You can update many options without regenerating the code:

- In the Business Hours block, you can:
  - change business hours of operation.
  - determine whether to terminate the call if it is outside business hours.

    - change the closed message (prompts).
- Update the Emergency block.
- Update prompts in the Menu block.
- Update prompts in the Play Message block.
- Specify the audio in the **Play Audio** tab of the Route Call block.
- Update the Special Day block.
- Update the input and retry prompts in the User Input block.

> **Tip**
> When nesting blocks, Genesys recommends that you do not go beyond ten (10) levels. Otherwise, you will receive a validation warning.

Block Categories

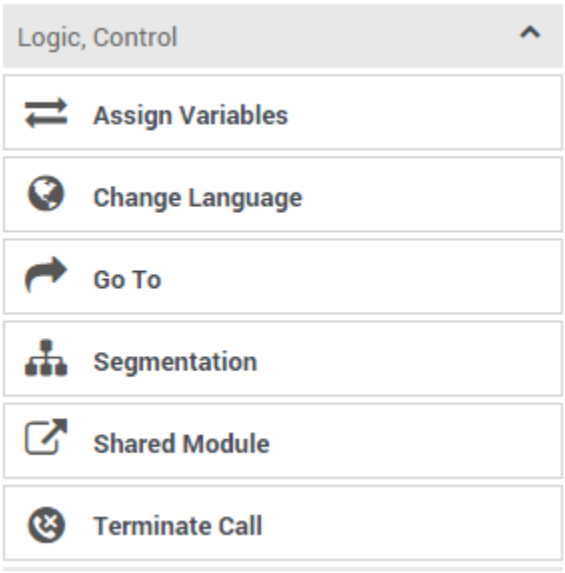
The blocks are grouped into the following categories:

- Logic and Control Blocks
- User Interaction Blocks
- Business Control Blocks
- Routing Blocks
- Data Blocks
- External Services Blocks
- Reporting Blocks
- Callback Blocks
- Survey Blocks

Logic and Control Blocks

The blocks in this category are used to add *logic* functions to an application, such as to assign variables, change the language (usually based on the caller's preference), and to provide *control* mechanisms within an application, such as to transition to another block, direct the application to follow a certain path, or end the call.

You might not see all of the blocks listed here on your Palette. The blocks shown depend on the features that are enabled and the type of application that is being built. For example, the **Terminate** block is only available for Digital application types.



Use the links below to learn more about each block.

Assign Variables

Assigns a new value or expression to user variables.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Change Language

Changes the language of the application and audio resources.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Go To

Enables transitions to other blocks.

Used in: **Self-Service**, **Assisted Service**

Return

Returns control from the Shared Module to the application or Shared Module that called it.

Used in: **Shared Modules**

Segmentation

Selects a path based on a specific runtime condition.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Shared Module

Splits larger applications into smaller pieces.

Used in: **Self-Service**, **Assisted Service**

Terminate Call

Disconnects the caller and stops the call.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Terminate (Digital only)

Ends the chat session.

Used in: **Assisted Service**, **Finalize**

Assign Variables Block

You can use the **Assign Variables** block in any phase of the application to assign a new value or expression to any of the user variables. Those variables can be used in other blocks whose properties support variables (for example, TTS prompts). The last-known state of variables is captured in metrics just before the SCXML session ends.

> **Tip**
>
> You specify user variables in the **User Variables** tab of the **Initialize** phase. When the application starts, those user variables are declared and assigned the user-specified default value.

You can use the **Sort Function** tab to sort the elements of a JSON array in a specified order. A maximum of three keys can be specified with each array. The same array can be sorted multiple times; therefore, the number of sort keys is unlimited.

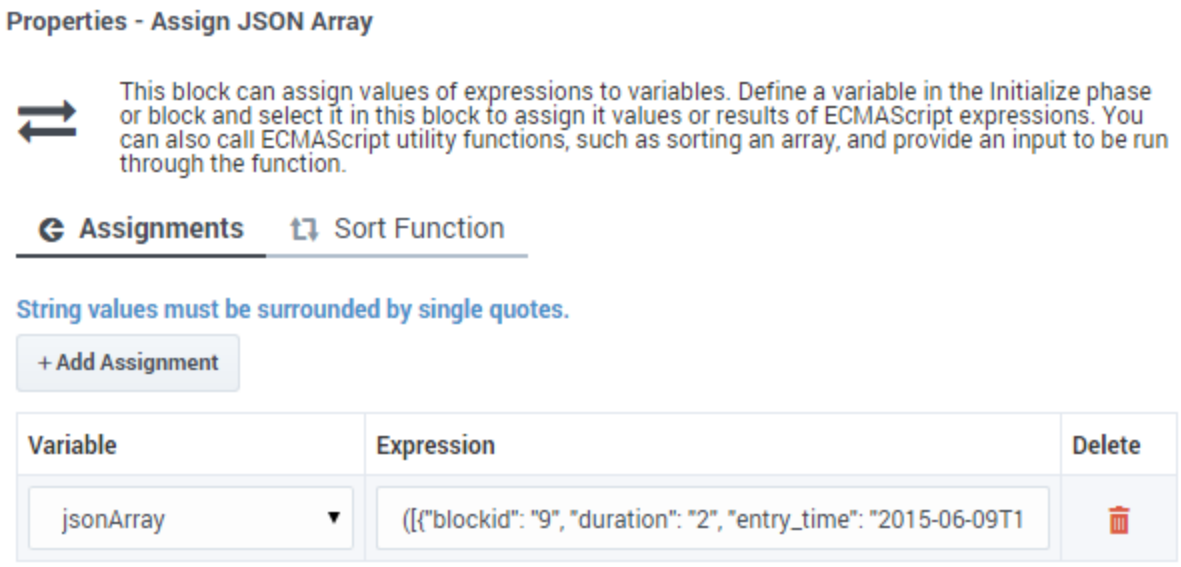
> **Important**
>
> The **Assignments** tab is processed before the **Sort Function** tab when your application executes the **Assign Variables** block. Do not assume that assignments are processed after sorting within the block. To extract specific parts of data after sorting, add another **Assign Variables** block after the one that performs sorting.

Assignments tab

Click **Add Assignment** to assign a value or expression to a variable.

- Select a variable from the **Variable** drop-down menu.
- Enter a value or expression for the variable in the **Expression** field. The value can be a simple literal value (such as a string, integer, or Boolean) or any valid JavaScript expression. The value expression can refer to other variables.

> **Tip**
>
> When assigning a string value to a variable, you must ensure that you enclose the string value with quotation marks. Otherwise, the string is interpreted as a reference to a variable.



Sort Function tab

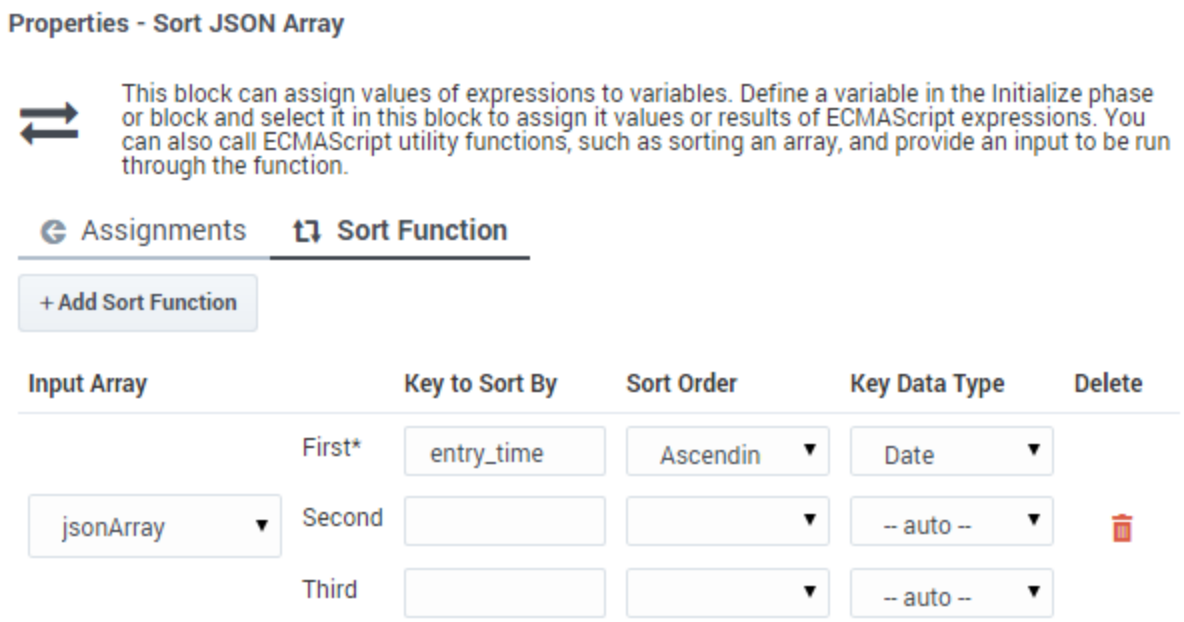
Click **Add Sort Function** and select an **Input Array** that contains a valid JSON array with values that you want to sort.

Enter up to three values in the **Key to Sort By** fields. These values must exist in the array. If an element does not have the specified value, it is skipped by the sort function and the value appears towards the end of the list.

Optionally, specify a **Sort Order** and **Key Data Type**. For the **date** type, you can use the following formats:

- *yyyy-MM-ddTHH:mm:ssZ*
  - Example in Greenwich Mean Time (Zulu): `2015-06-01T12:13:14Z`
- *yyyy-MM-ddTHH:mm:ss[+-]HHmm*
  - Example in Pacific Time: `2015-06-01T12:13:14-0800`

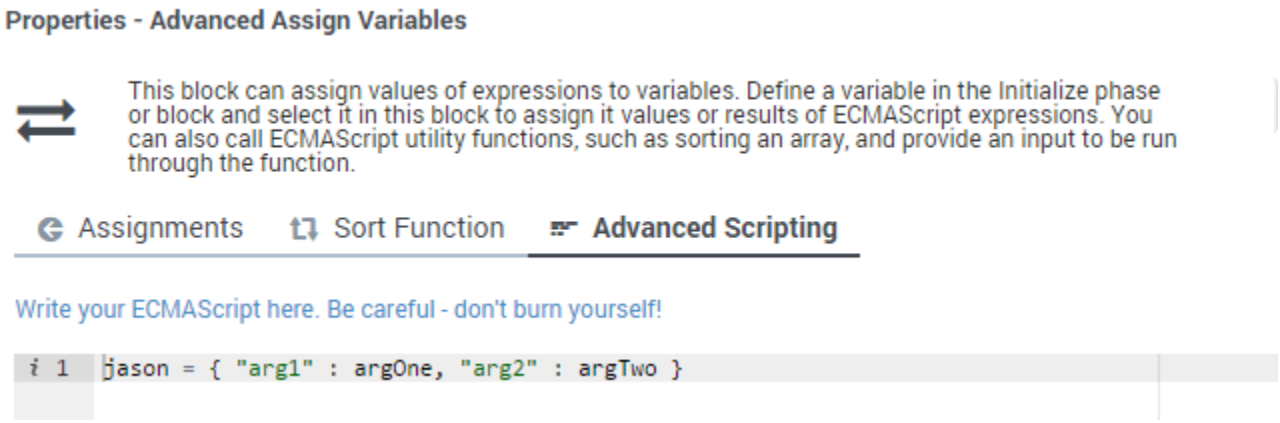
> ### Tip
> You can specify a **Key Data Type** to use data-aware sorting to treat different keys differently. If you do not specify a data type, the sort function treats all sort keys as strings and sorts those strings.



Advanced Scripting tab

> **Important**
>
> Advanced Scripting is an optional feature and might not be enabled on your system. To enable this functionality, contact Genesys.

Click **Advanced Scripting** to enter your own ECMAScript expression.



Change Language Block

You can use the **Change Language** block to change the language of the application. This also changes the language in which audio resources are played.

Typically, you use this block to switch languages once the caller's language preference is determined. This may be determined by prompting the caller to select his preferred language, using logic in the application (for example, a call that is routed to a regional contact center might use a default language setting for each region), or a RESTful API call into a customer preferences database that returns the preferred language.

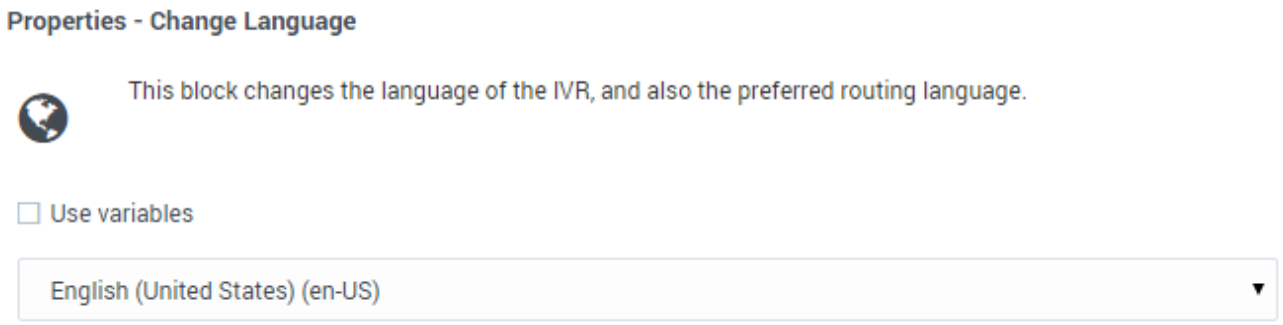
You can use the **Change Language** block in any phase of the application.

Important

- If a Self Service shared module called from the Self Service phase of the application changes a language, that language stays in effect when the module returns to the calling flow.
- You must upload audio resources for all languages used in the application before the application is run.
- If you set the language of an application to a variable, you must ensure that the associated audio files have been uploaded before the application is run. Otherwise, if these audio files are missing, no audio is available to play during the call. Designer cannot detect this error when you click **Publish** to validate and save your application.

Using this Block

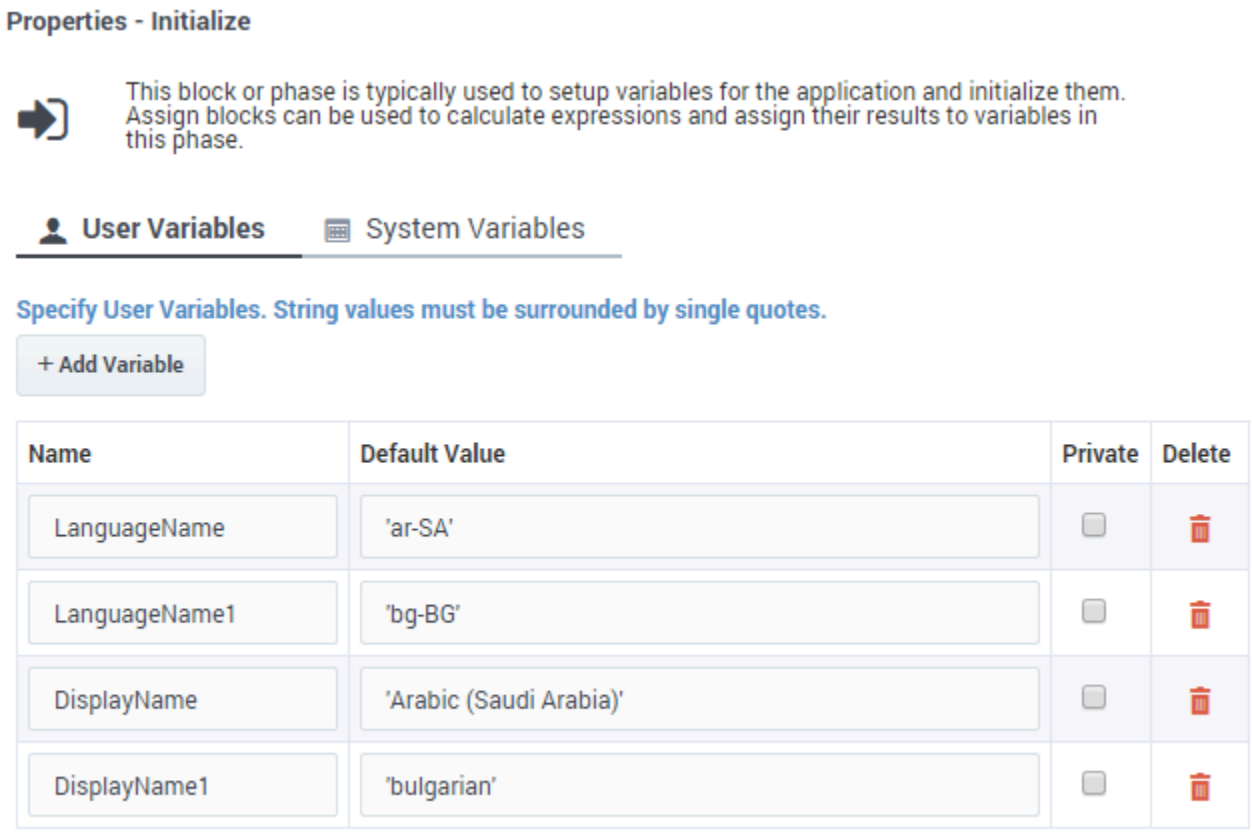
You can select a language from the drop-down menu:



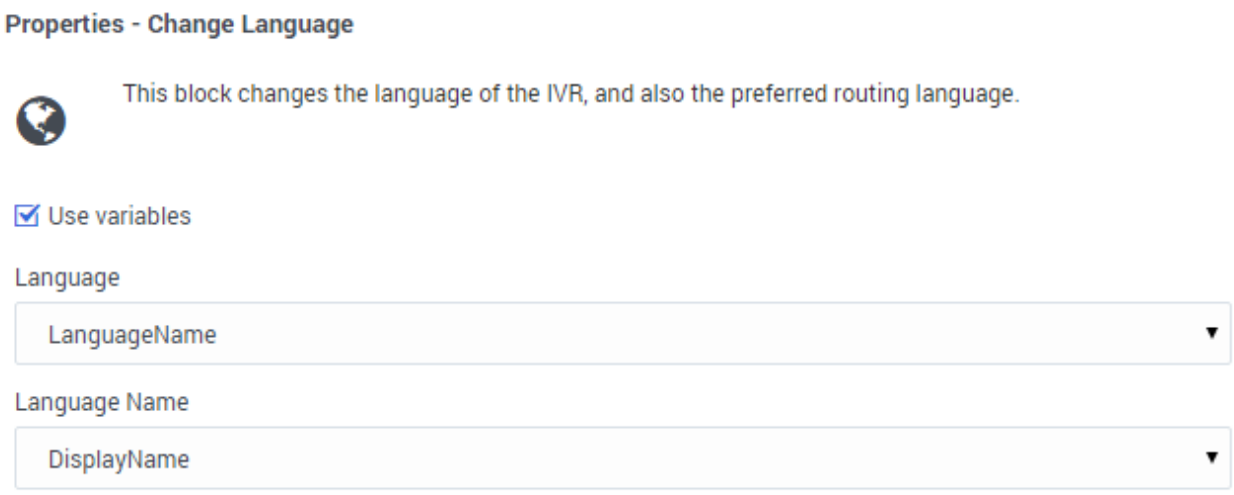
Or, you can select the **Use variables** check box to specify variables for the Language and Language

Name. Here's an example of how to do this:

- First, specify your language variables in the **Initialize** phase:



- Then, select them in the **Change Language** block:

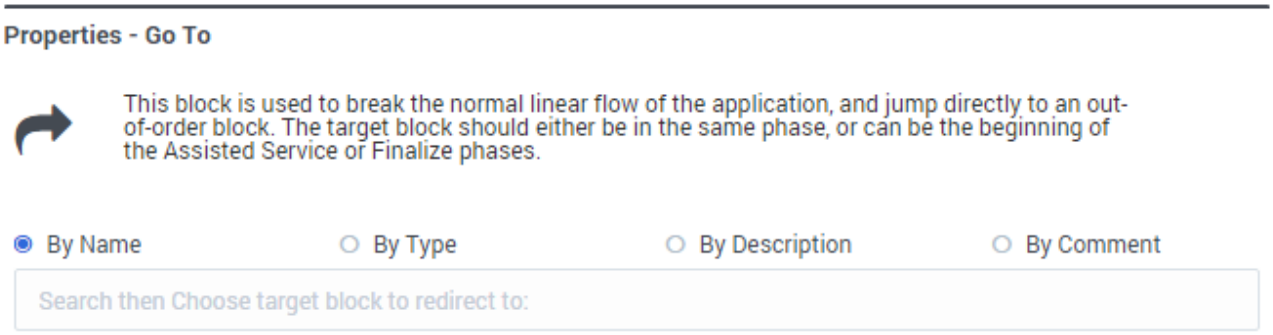

Properties - Change Language
This block changes the language of the IVR, and also the preferred routing language.
Use variables
Language
LanguageName
Language Name
DisplayName

Go To Block

You can use the **Go To** block to enable transitions to other blocks in the same phase of an application or to the beginning of the **Assisted Service** or **Finalize** phase. You cannot use the **Go To** block to transition directly to **Menu Option** or **Segmentation Option** blocks.

Using this Block

Use the radio buttons to select a search criteria (for example, by **Name**, **Type**, **Description**, or **Comment**), then start entering the term you are looking for. Designer starts returning the results as you type.



Return Block

The **Return** block is available only for Shared Modules and is used to return control from the Shared Module to the application or Shared Module that called it. Multiple **Return** blocks can be used in a single Shared Module.

This block can return values of any output variables from the Shared Module. In the **Initialize** phase, variables can be marked as **Output** variables that are expected to be returned from this Shared Module. Only those variables can be assigned return values in the **Return** block.

Using this Block

Click **Add Assignment** and specify output variables.

Segmentation Block

You can use a **Segmentation** block to take a different path depending on the specific values of application variables. A valid ECMAScript expression containing application variables, ECMAScript operators, and Designer functions can be used to define a Segmentation Option. If this condition is evaluated to a *true* (Boolean) value while the application executes, the application flow takes the path of that Segmentation Option.

You can define multiple Segmentation Options, each with their own conditions. For example, the condition can be a variable with a Boolean value, a call to a function that returns a Boolean, or a combination of variables with logic operators that evaluates to a Boolean.

The first condition that evaluates successfully is selected as the segmentation path, and any blocks under that Segmentation Option are executed. If no condition expression evaluates successfully, none of the Segmentation Options execute, and the application executes the block that follows the **Segmentation** block.

Application variable values can be set based on logic in the application, by querying external data sources from blocks (such as the HTTP REST block), or by collecting input from a caller in the User Input block.

Conditions are ordered and exclusive, which means:

- Condition expressions are evaluated in the order they are defined.
- If one condition evaluates to true and the corresponding path is selected, then the following condition expressions are not tested. After executing this segment path, the application executes the block that follows the **Segmentation** block.

> ### Tip
> If the same logic needs to be executed in multiple segmentation paths, Genesys recommends that you keep the paths for each option independent and avoid using **GoTo** blocks to jump between paths. The common logic can be moved into a Shared Module, which can then be called from multiple paths. This improves the structure and reliability of your application.

The **Segmentation** block selects the first segment whose condition is a valid ECMAScript expression that evaluates to *true* (Boolean). If none of the conditions evaluate to *true*, no segment is executed, and processing moves on to the next sibling of the **Segmentation** block.

> ### Warning
> You must use condition expressions that evaluate to a Boolean value. Expressions that evaluate to a different data type can result in errors.

The following are valid expressions:

- Using a variable whose value is *true* or *false* and comparing it to a Boolean value, such as the variable used to hold the result of a **Special Days** block:

  ```
  isSpecialDayVar == true
  ```

  or:

  ```
  isSpecialDayVar == false
  ```

- Using a Boolean property of an object stored in a variable, such as the **Route Call** block outcome variable:

  ```
  routeCallOutcomeVar.success == true
  ```

- An expression using Boolean variables and logical operators (&&, ||):

  ```
  var1 == false || (var2 == true && var3 == true))
  ```

- An expression using comparison operators (==, ===, !=, !==, >, <, >=, <=):

  ```
  var1.length > 3 || var2 === 'stop'
  ```

> **Important**
>
> Do not use an expression that does not have explicit comparison operators (such as `varIsHoliday`).

When using condition expressions that do not evaluate to a Boolean value, the following rules apply:

> **Important**
>
> - These rules are general ECMAScript/Javascript rules, and apply as-is to Designer. (This not the regular "flavor" of Javascript that runs in a browser. Instead, this Javascript is executed by Genesys platform components and has certain restrictions.)
> - It is mandatory to ensure these expressions evaluate to a Boolean value and not to other data types or values, such as *undefined*.

- The condition expression evaluates to an object => the condition is considered *true* (this applies to arrays, even if they are empty). Instead, use the following:

  ```
  typeof myVar === 'object'
  ```

- The condition expression evaluates to *undefined* => the condition is considered *false*. Instead, use the following:

  ```
  myVar !== undefined
  ```

- The condition expression evaluates to *null* => the condition is considered *false*. Instead, use the

following:

```
myVar !== null
```

- The condition expression evaluates to a number => the condition is considered *false* if the value is `+0`, `-0`, or `NaN`; otherwise, the condition is considered *true*. Instead, use the following:

  ```
  myVar === 3
  ```

- The condition expression evaluates to a string => the condition is considered *false* if the string is empty; otherwise, the condition is considered *true*. Instead, use the following:

  ```
  myVar.length > 0
  ```

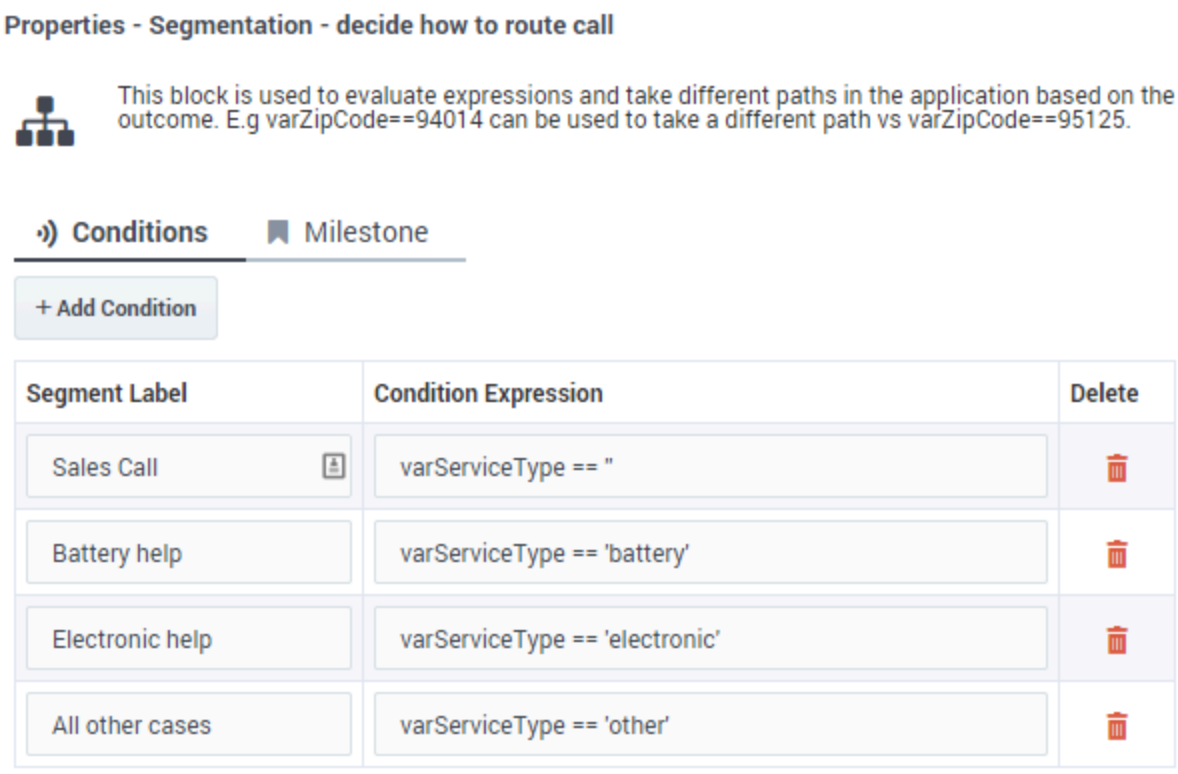
Conditions tab

Click **Add Condition** and type the condition to evaluate in the **Condition Expression** field. The value can be a simple Boolean value, a variable with some Boolean content, or any valid JavaScript expression that evaluates to a Boolean. The condition expression can refer to other variables.

You can edit the **Segment Label** field to give a meaningful label to your segment. The child segment block will be named accordingly.

To remove a condition, click the trash icon for that condition in the **Segmentation** block or click the trash icon on the related child block.

> ### Important
> Always make sure the condition evaluates to a Boolean value at runtime.



Milestone tab

Add a Milestone to mark this key moment while the application is running, similar to within the Milestone block.

Shared Module Block

Shared Modules are useful for reusing code from multiple applications, as well as for splitting larger applications into smaller and more manageable chunks. Once you have created a Shared Module, you can use the Shared Module block to invoke the module into your application.

If you change a Shared Module, you also change all of the applications that use that module. If an application uses the **Latest** version of a module, and the application is published, it starts using the new state of the Shared Module. If an application uses a specific version of the Shared Module (not the **Latest**), it does not receive the latest changes in the Shared Module, even if the application is published again.

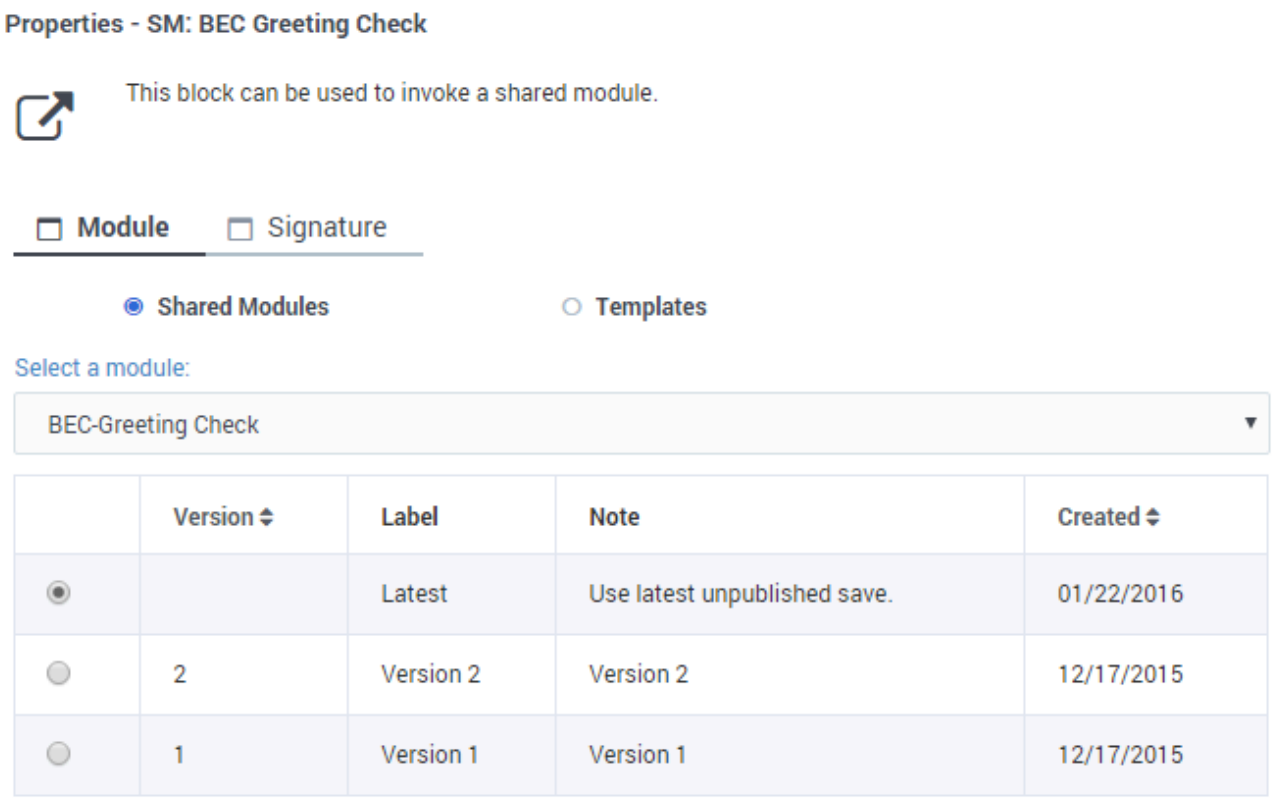
For more information about how to create and manage shared modules, see Shared Module.

Module tab

Select a Shared Module or Template.

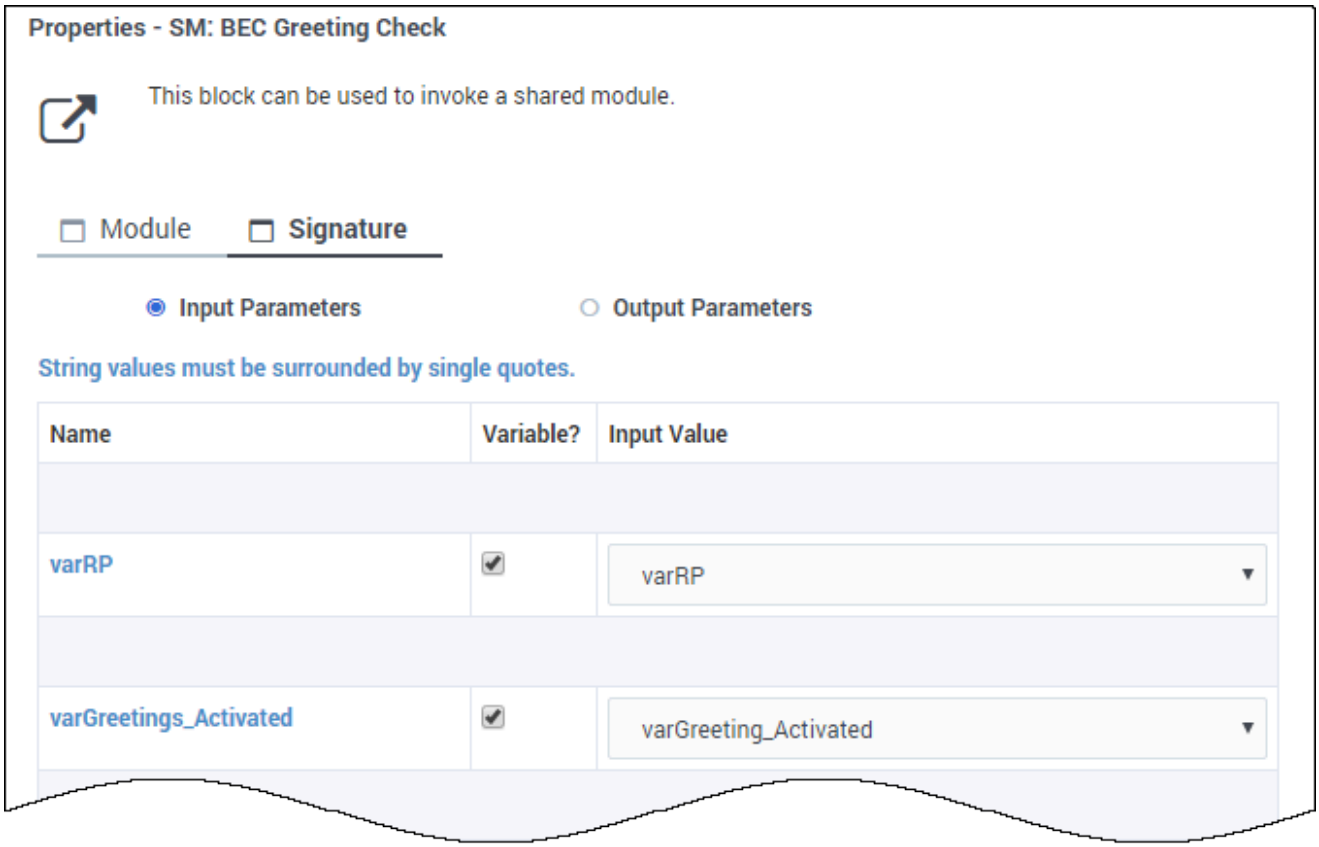
All Shared Modules that have at least one version are listed. Once a Shared Module is selected, all published versions of the module are shown. Usually the latest version should be selected, unless there is a incompatibility with the latest version.

Templates are used only with the Callback block. They are read-only and cannot be edited or deleted.



Signature tab

Specify the values for the **Input** and **Output Parameters**. You can use literals, variables, or expressions.


Properties - SM: BEC Greeting Check
This block can be used to invoke a shared module.
Module
Signature
Input Parameters
Output Parameters
String values must be surrounded by single quotes.
Name
Variable?
Input Value
varRP
varRP
varGreetings_Activated
varGreeting_Activated

Terminate Call Block

You can use the **Terminate Call** block to disconnect the caller and stop the call. Everything after the **Terminate Call** block is skipped, and the application moves straight to the **Finalize** phase.

As a visual aid, the right edge of the **Terminate Call** block is capped in red, to show that the application will stop if and when it reaches this block. This visual aid also applies to any block that might end the call, such as Business Hours or Special Day, when the Terminate Call option is enabled.

Terminate Block (Digital)

You can use the **Terminate** block in a Digital application type to end an interaction.

When used in the **Self Service** or **Assisted Service** phases, everything after the **Terminate** block is skipped and the application moves straight to the **Finalize** phase.

When used in the **Finalize** phase, the application sets the termination flag and moves to the next block.

As a visual aid, the right edge of the **Terminate** block is capped in red to show that the application will stop if and when it reaches this block. This visual aid also applies to any block that might end the call, such as Business Hours or Special Day, when the Terminate option is enabled.

User Interaction Blocks

The blocks in this category are used to *interact* with callers in various ways, such as to offer them a list of menu options ("Press 1 to speak with an agent"), collect their information (such as an account number), play them a message, or record their call (or a selected portion of the call).

The blocks shown depend on the features that are enabled and the type of application that is being built. For example, only Digital applications will see blocks related to Chat.

Use the links below to learn more about each block.

Menu

Presents a list of choices to callers.

Used in: **Self Service**

Play Message

Plays audio messages to callers.

Used in: **Self Service**, **Assisted Service**

Record

Starts or stops a call recording.

Used in: **Self Service**

Record Utterance

Records a user's voice or DTMF inputs.

Used in: **Self Service**

User Input

Collects information from callers.

Used in: **Self Service**, **Assisted Service**

Chat Message (Digital only)

Sends a chat message to a contact.

Used in: **Assisted Service**

Chat Transcript (Digital only)

Emails the chat transcript to a contact.

Used in: **Assisted Service**, **Finalize**

Get Chat Transcript (Digital only)

Provides access to the latest chat transcript.

Used in: **Assisted Service**, **Finalize**

Send Email (Digital only)

Sends an email containing a standard-response message to a user.

Menu Block

You can use the **Menu** block only in the **Self Service** phase to present a list of choices to the caller and accept a selection that the caller provides by using a DTMF key press.

You can choose to enable certain DTMF keys and associate specific processing or logical flow with those keys. For each DTMF key that is enabled, a new **Menu Option** block is shown in the **Application Flow**. You can then add new blocks to each of these **Menu Option** blocks.

DTMF Options tab

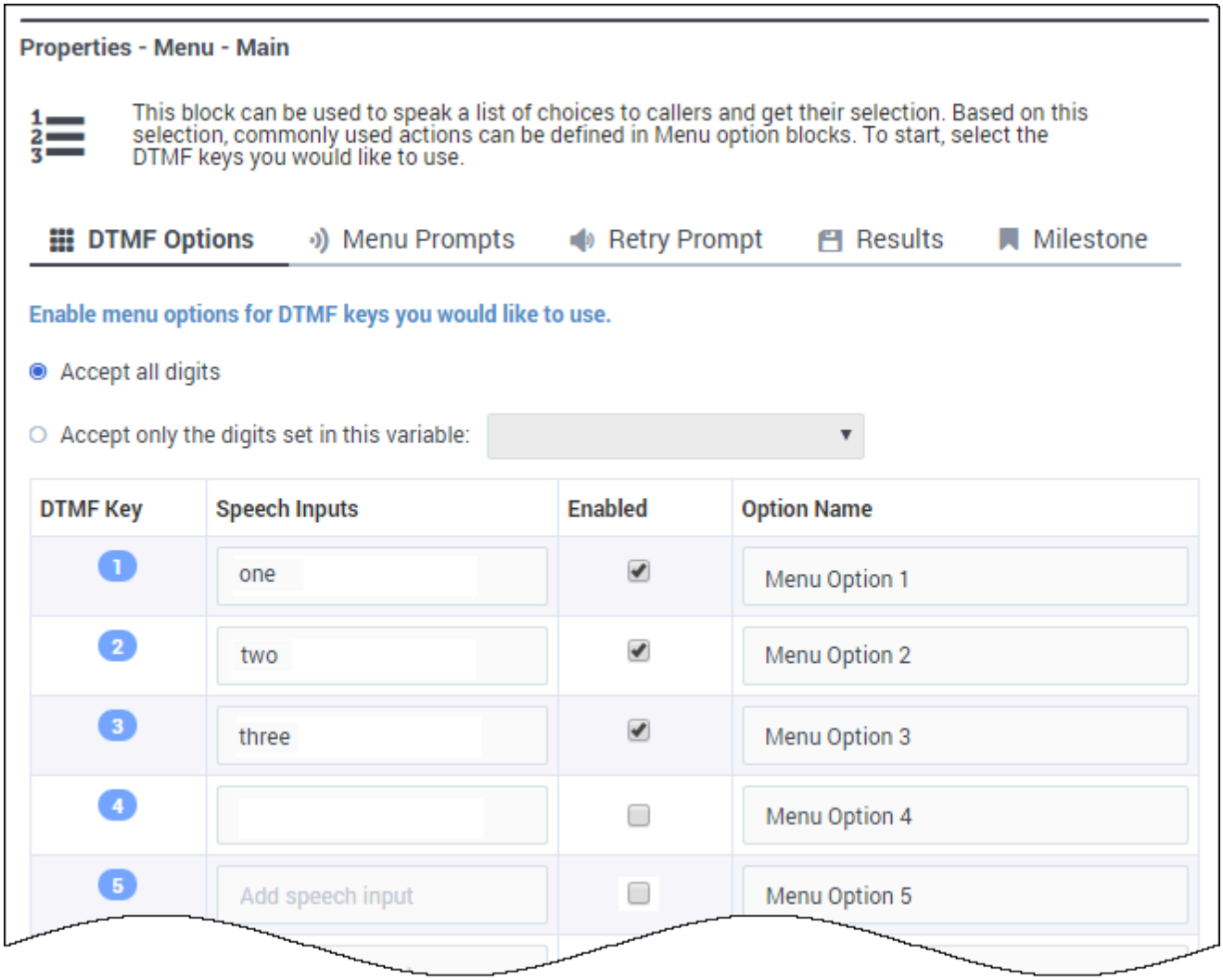
Select one or more DTMF keys, which enables a **Menu Option** block for each key.

Select **Accept all digits** or **Accept only the digits set in this variable**. Using the variable option allows you to set conditions for enabling or suppressing specific menu options while the application is running.

Use a descriptive **Option Name** to make it is easier to understand the flow.

Optionally, enter a valid speech input for each DTMF key in the **Speech Inputs** field.

Refer to the Menu Option block page for more information on how to configure **Menu Option** blocks.



Menu Prompts tab

Input timeout

Specify the number of seconds that the application should wait before assuming that no input was received. The default value is 5 seconds.

Disable barge-in

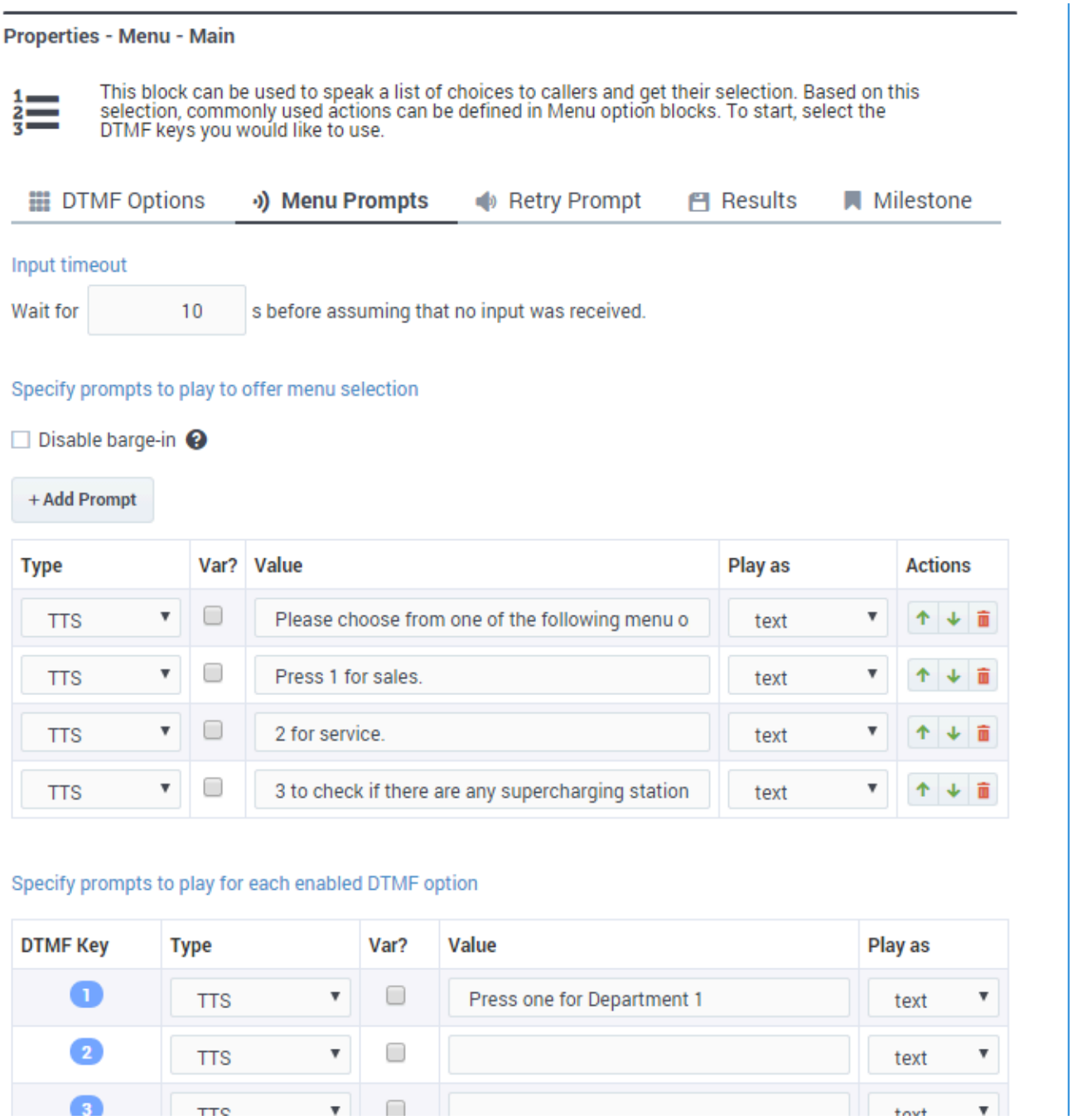
Select this option to prevent callers from interrupting a prompt while it is still playing. For example, you might want a "Welcome" message to play all the way through before the caller can enter another command and skip to the next menu prompt.

If this option is not selected, barge-in is enabled, and the prompt can be interrupted by the caller.

> **Important**
>
> The selected barge-in setting applies irrespective of whether global DTMF commands are used or not.

Click **Add Prompt** to play prompts when the menu starts.

> **Tip**
>
> See the Play Message block page for more information on how to create prompts.



Retry Prompt tab

Allow Retries

Select to allow callers to provide late input or an unrecognized input. If enabled, you can set the

following options:

- **Number of No Input retries allowed**
  Enter the number of retries to allow for callers whom do not provide input. For each retry, you can specify whether a prompt is played by clicking the corresponding section beneath this field. For example, if you allow two no-input retries and you want to play a prompt after the first retry, select the **No Input #1** line and add a prompt. Enable the **Play original menu prompt after this retry prompt** check box to repeat the menu prompts for the caller.
- **Number of No Match retries allowed**
  Enter the number of retries to allow for callers whom do not provide a match for a **Menu Block**. For each retry, you can specify whether a prompt is played by clicking the corresponding section beneath this field. For example, if you allow two no-input retries and you want to play a prompt after the first retry, select the **No Match #1** line and add a prompt. Enable the **Play original menu prompt after this retry prompt** check box to repeat the menu prompts for the caller.
- **After Final No Input**
  Add the prompt to play after the maximum number of permitted No Input retries is reached. You can also specify a target destination for the application to jump to, such as another block in the Self Service phase or to the Assisted Service or Finalize phase of the application.
- **After Final No Match**
  Add the prompt to play after the maximum number of permitted No Match retries is reached. You can also specify a target destination for the application to jump to, such as another block in the Self Service phase or to the Assisted Service or Finalize phase of the application.

Results tab

Select variables to store the user's DTMF selection and the outcome of the interaction.

Milestone tab

Add a milestone to mark this key moment while the application is running. See the **Milestone** block page for more information.

Menu Option Block

Menu Option blocks appear in the **Application Flow** after you enable at least one DTMF key in a Menu block.

> **Important**
>
> When configuring Menu block options, Genesys recommends that you keep the branches of each option independent and use Shared Modules to share any functionality between them (rather than pointing to the child block of another option within the same branch). This improves the efficiency and reliability of your application.

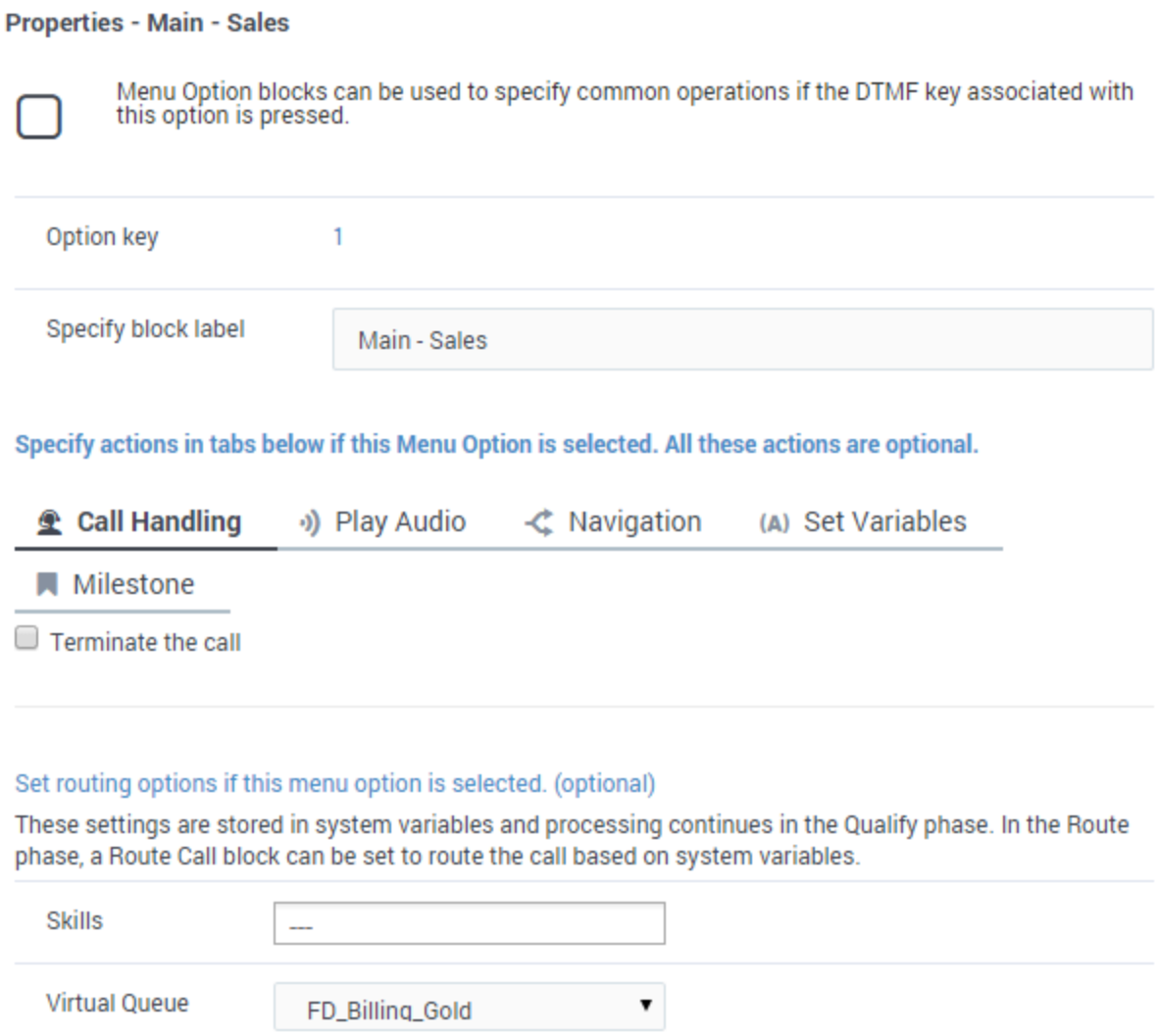
Call Handling tab

Terminate the call

Enable this option to terminate the call if this menu option is selected by the caller.

Optionally, you can choose to route this call if this menu option is selected by the caller. If so, select a Skill and Virtual Queue to which the call will be routed. These selections are stored to the **RoutingSkills** and **RoutingVirtualQueue** system variables, respectively.

> **Important**
>
> If you set these routing options, Designer does not route the call unless a Route Call block is added to the **Assisted Service** phase that routes based on menu options.



Play Audio tab

Disable barge-in

Select this option to prevent callers from interrupting a prompt while it is still playing. For example, you might want a "Welcome" message to play all the way through before the caller can enter another command and skip to the next menu prompt.

If this option is not selected, barge-in is enabled, and the prompt can be interrupted by the caller.

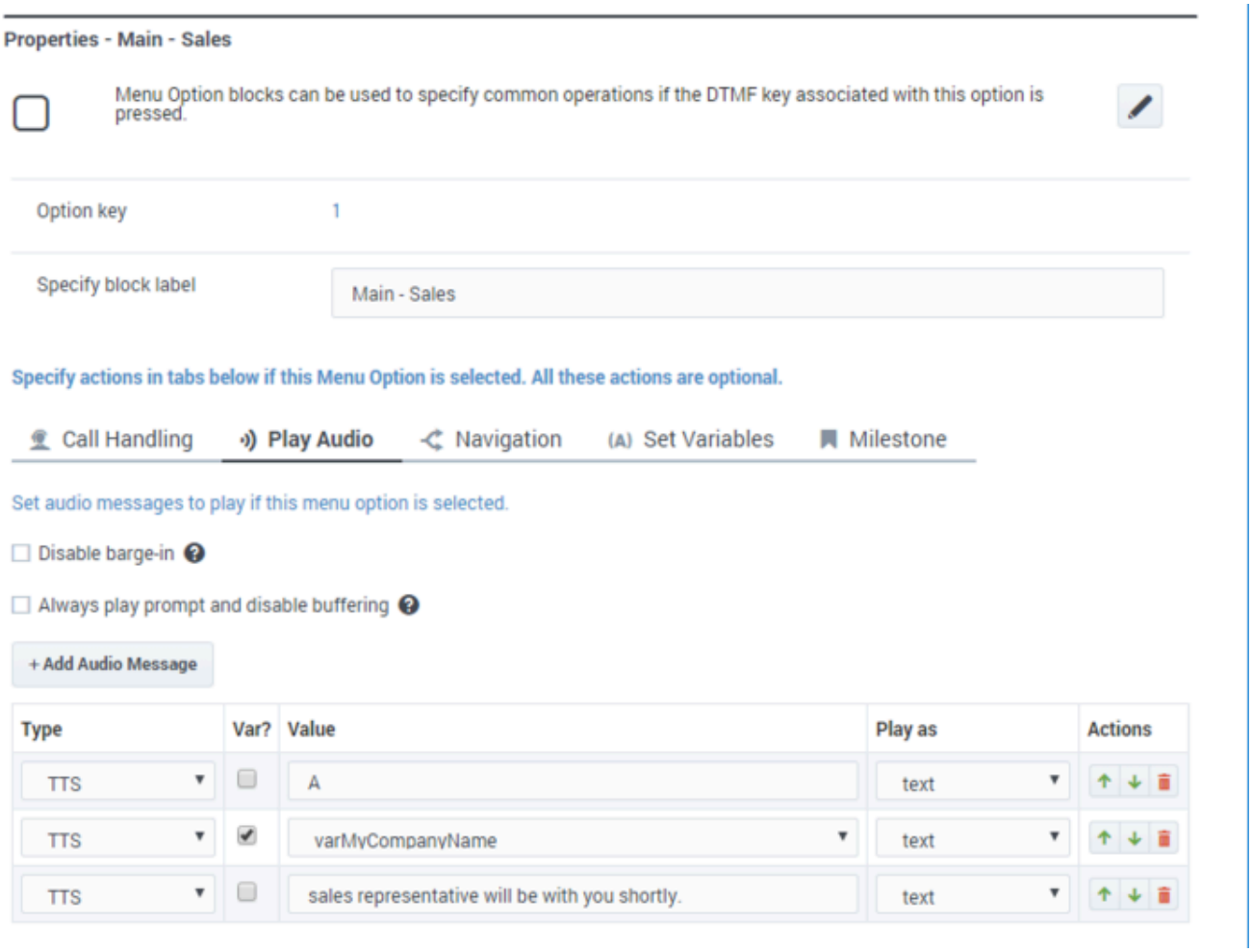
> **Important**
> The selected barge-in setting applies irrespective of whether global DTMF commands are used or not.

Always play prompt and disable buffering

Select this option if you want callers to be able to interrupt a prompt while it is playing, but not have those inputs applied to subsequent User Input or Menu block prompts. For example, if this option is enabled and the caller interrupts a “Welcome” message by pressing 3, the input is ignored by the next User Input or Menu prompts.

If this option is not enabled, the input is buffered and applied to the next block accepting input.

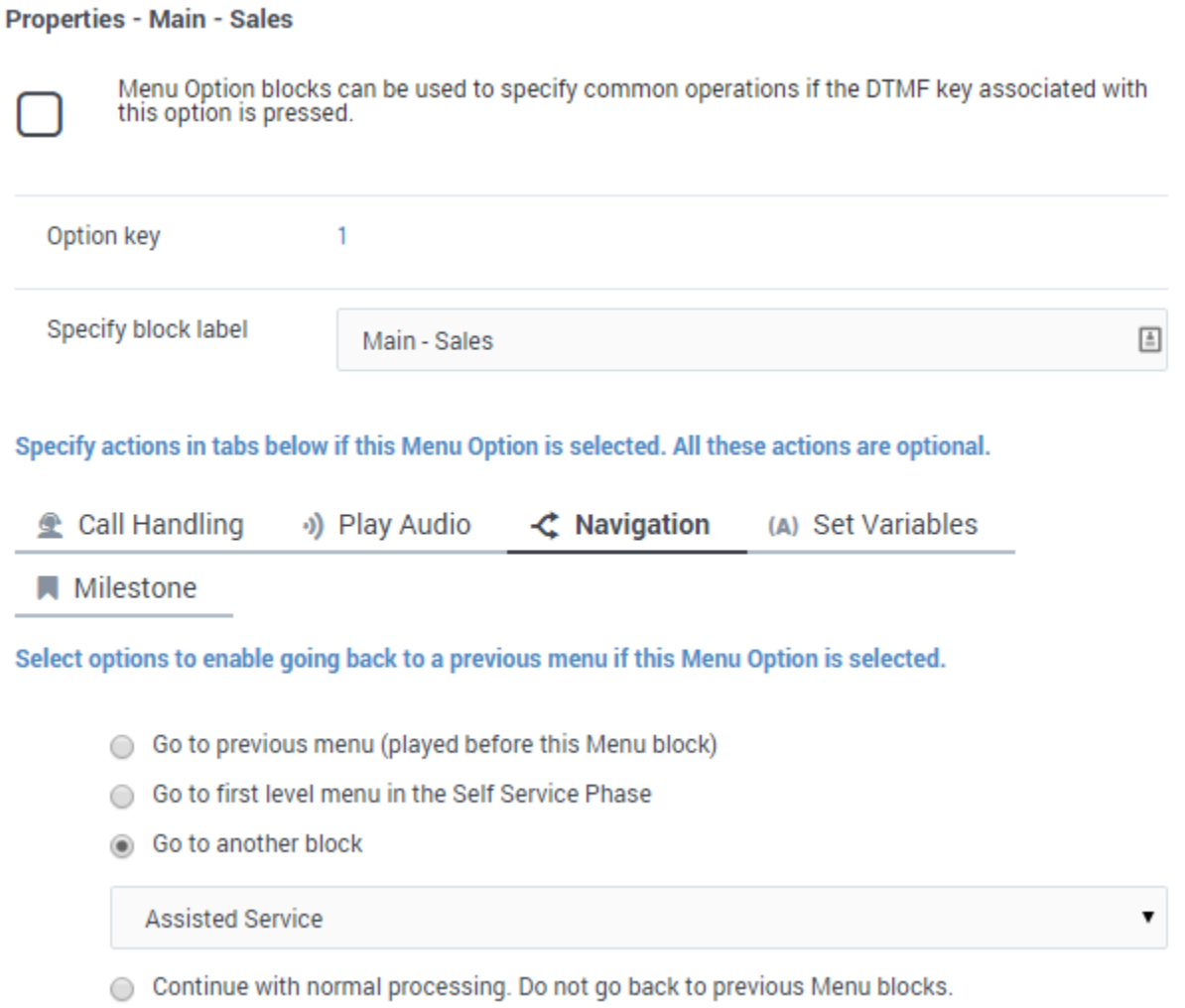
Click **Add Audio Message** to play audio if this specific menu option is selected.



Navigation tab

Select where the application proceeds after this menu option is selected by the caller.

> **Tip**
> If there are hierarchical menus in your application, it is a good idea to provide callers with an option to go to a previous menu.



Set Variables tab

Assign variables to use when this menu option is selected by the caller, without having to add an

Assign Variables block.

Milestone tab

Add a milestone to mark this key moment while the application is running. See the Milestone block page for more information.

Play Message Block

You can use the **Play Message** block in the **Self Service** and **Assisted Service** phases to play audio messages to the caller. These messages or prompts might be an introductory welcome message or instructions on how to proceed through the application.

These audio messages are defined as either:

- Text-to-Speech (TTS) — Strings entered directly in the block, or variables.
- Announcements — Audio files that were previously uploaded in the Audio Resources page, or variables played as TTS.

Using this Block

Disable barge-in

Select this option to prevent callers from interrupting a prompt while it is still playing. For example, you might want a "Welcome" message to play all the way through before the caller can enter another command and skip to the next menu prompt.

If this option is not selected, barge-in is enabled, and the prompt can be interrupted by the caller.

> **Important**
>
> - The selected barge-in setting applies irrespective of whether global DTMF commands are used or not.
> - This option is only supported if the **Play Message** block is used during the **Self Service** phase. For **Assisted Service**, you can use the User Input block to control barge-in settings.

Always play prompt and disable buffering

Select this option if you want callers to be able to interrupt a prompt while it is playing, but not have those inputs applied to subsequent User Input or Menu block prompts. For example, if this option is enabled and the caller interrupts a “Welcome” message by pressing 3, the input is ignored by the next User Input or Menu prompts.

If this option is not enabled, the input is buffered and applied to the next block accepting input.

> **Important**
>
> This option is only supported if the **Play Message** block is used during the **Self Service** phase.

To create a new prompt, click **Add Prompt** and follow the instructions below.

In the **Type** column, select the type of prompt:

- **TTS** — Read a text or variable value to a user through TTS.
- **Announcement** — Play a prerecorded announcement. When using a variable, the variable value should be the name of the audio resource to play.
- **Intelligent Prompt** — Intelligently convert a number into items such as a date, currency, or ordinal number, and then read it with human audio to a user.

> **Important**
>
> If Designer is not able to play an Intelligent Prompt in the caller's preferred language, it will play the prompt in American English (en-US).

In the **Variable?** column, enable or disable the check box to identify the **Value** as a variable.

In the **Value** column, specify the prompt value. If **Variable?** is enabled, choose a variable in the drop-down menu.

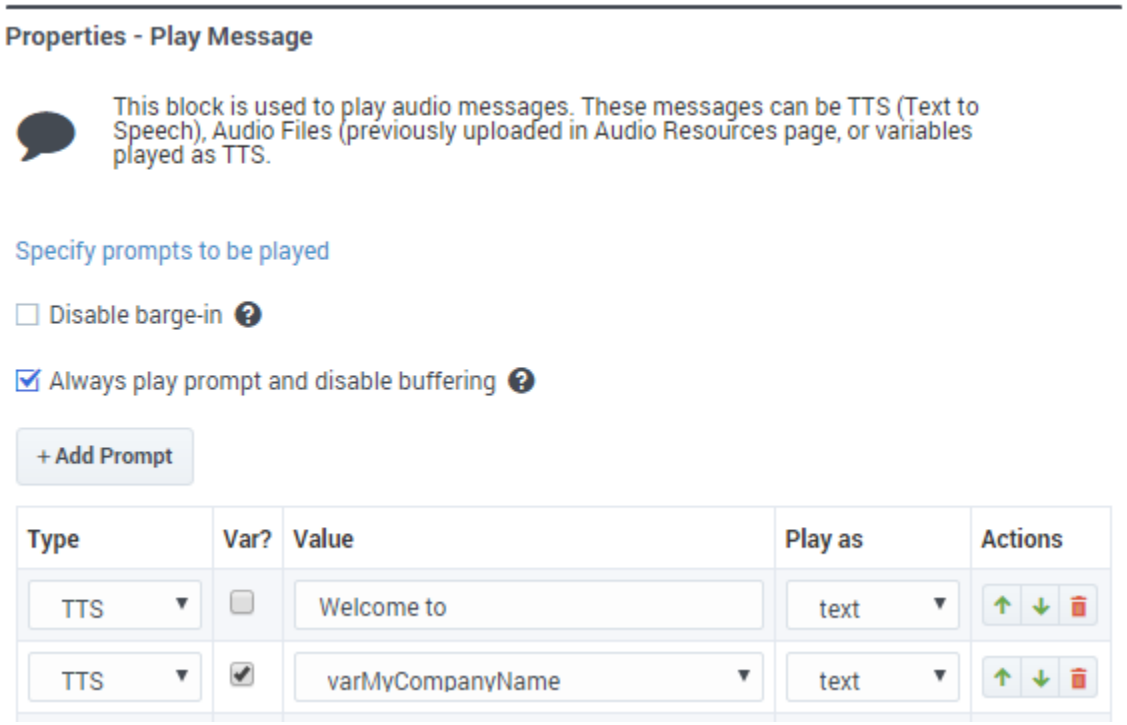
In the **Play as** column, select an option:

> **Important**
>
> Some **Play as** options might not be available for certain prompt types.

- **alphanumeric** - The value is read as a series of letters and/or numbers.
- **currency** — Use the following format: `UUUMM.NN`, where `UUU` is the ISO4217 currency code. You can omit the currency code to use the default currency for the current locale.
- **date** — Use the following format: `YYYYMMDD`. You can use `??` or `????` for unspecified fields.
- **day** - A day of the week.
- **dtmf** - A menu item.
- **ordinal** — A positive integer.
- **cardinal** — A positive or negative integer or decimal number.
- **character** - A character.

- **text** - Text that should be read without special formatting (for example, a sentence or phrase).
- **time**
  - TTS prompt - You must use the following format: `hh:mm`. For example, use `09:00` for 9 a.m. or `21:00` for 9 p.m.
  - Intelligent Prompt - You can use the TTS format or the following format: `hhmm[aph?]`, where a is a.m., p is p.m., h identifies 24-hour time, and ? is unspecified. For example, you can use `0900` for 9 a.m. or `0900p` for 9 p.m.
- **telephone** or **phone** — Use a sequence of digits (`0` - `9`), optionally followed by an "x" and then extension digits (`0` - `9`).

Example



Scenarios

If you want to repeat the account number that the caller just entered:

1. First Prompt
   - **Type**: TTS
   - **Variable?**: Disabled
   - **Value**: `The account number you just entered is`

- **Play as**: text

2. Second Prompt
   - **Type**: TTS
   - **Variable?**: Enabled
   - **Value**: account_number_variable
   - **Play as**: telephone

If you want to allow barge-in on a "Welcome" message, followed by an informational prompt for a Menu block input that has barge-in and buffering disabled:

1. In the properties for the **Play Message** block for the "Welcome" message:
   - Do not select **Disable barge-in**.
   - Select **Play prompt and disable buffering**.
2. In the properties for the Menu block that prompts for the caller's input:
   - Select **Disable barge-in**.

Record Block

You can use the **Record** block to control a call recording during the **Self Service** phase of an application.

After recording is started, all interactions are recorded until the specified stop setting is reached, the **Self Service** phase ends, or the call is completed.

The **Record** block is useful when you want to

- record the entire IVR call flow but avoid capturing any sensitive or private information (such as a caller's SSN or credit card number) — you could stop recording just before the caller is asked for personal information, and then start it afterward.
- record only a specific part of a call, such as when a customer is asked to confirm or agree to make a payment — you could start recording for the part of the call flow where the customer is asked for this information, and then stop it afterward.

After the recording stops, the platform mixes the input and output audio channels into a single audio file.

Things you should know

Depending on whether your environment uses **Full Call Recording (FCR)** or **Genesys Interaction Recording (GIR)**, the **Record** block might not support all of the features or options described on this page (in most cases, you won't see an option on your screen if it isn't supported).

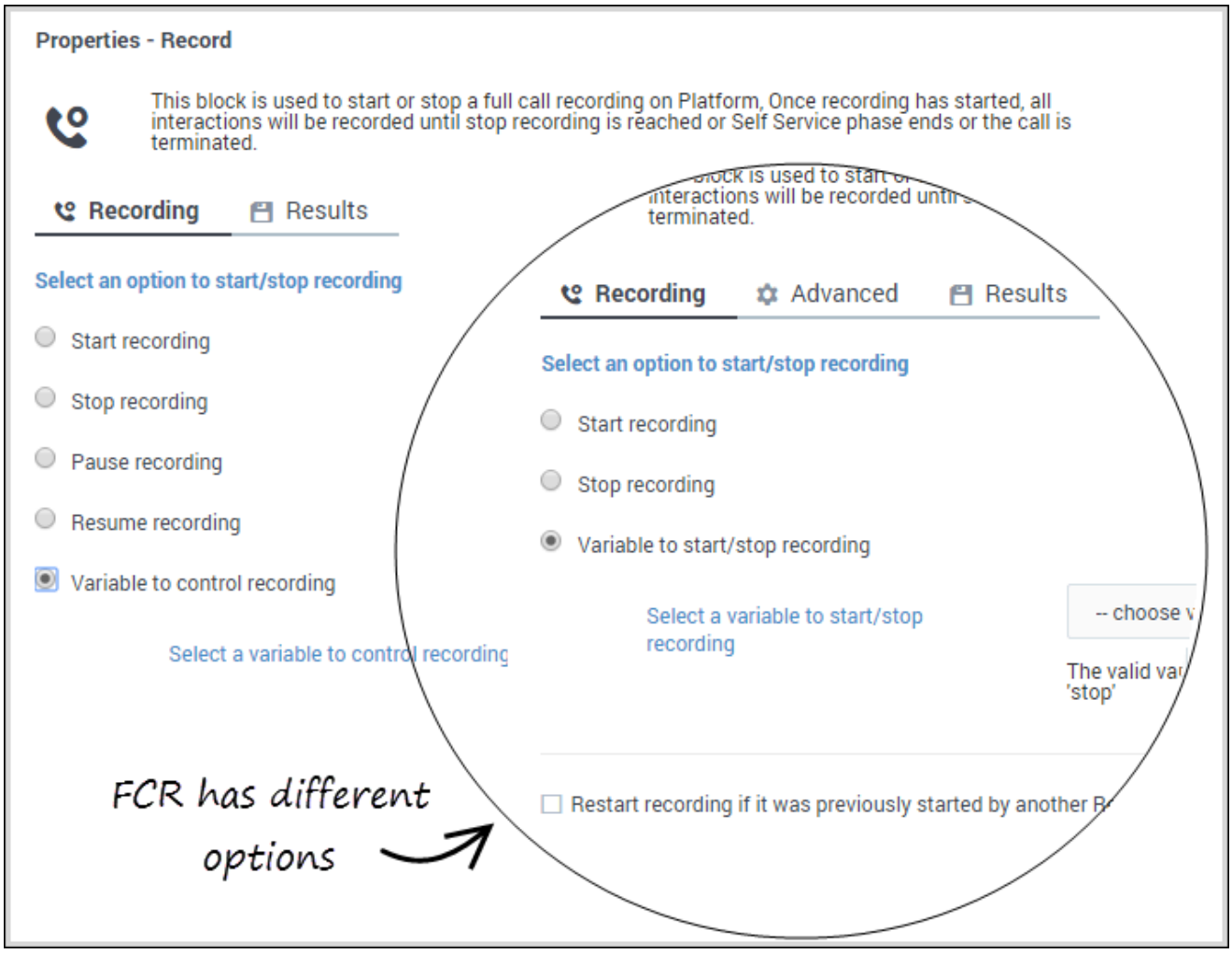
The Designer instance can only support one type of call recording. If you are not sure which recording type applies to your site, please check with Genesys.

If your recording type has changed, you might also need to review Migrating Applications for information about migrating your existing applications to the new recording type.

> **Important**
>
> If you are using FCR and have enabled call recording using the *EnableSSRecording* variable, the application ignores the **Record** blocks.

Recording tab

Select an option to **start** or **stop** the recording, or choose a variable. If your environment uses GIR, you will also have additional options available to **pause** or **resume** the recording.



Select the **Variable to start/stop recording** when you need to start or stop the recording dynamically based on a condition during run-time.

If you are using GIR, this option appears as **Variable to control recording**, and includes additional options to pause or resume the recording.

You would then select a user variable that will be assigned the value of the recording action you want, depending on the condition experienced during the application flow (such as to *stop* recording if a caller chooses a certain menu option).

The recording is controlled based on the value assigned to the variable. If the variable does not contain a valid value, the application skips the block and continues with the flow.

Select the **Restart recording if it was previously started by another Record block** option if you want to start a new recording file instead of continuing with the recording file that was already started. (This option isn't available for environments using GIR.)

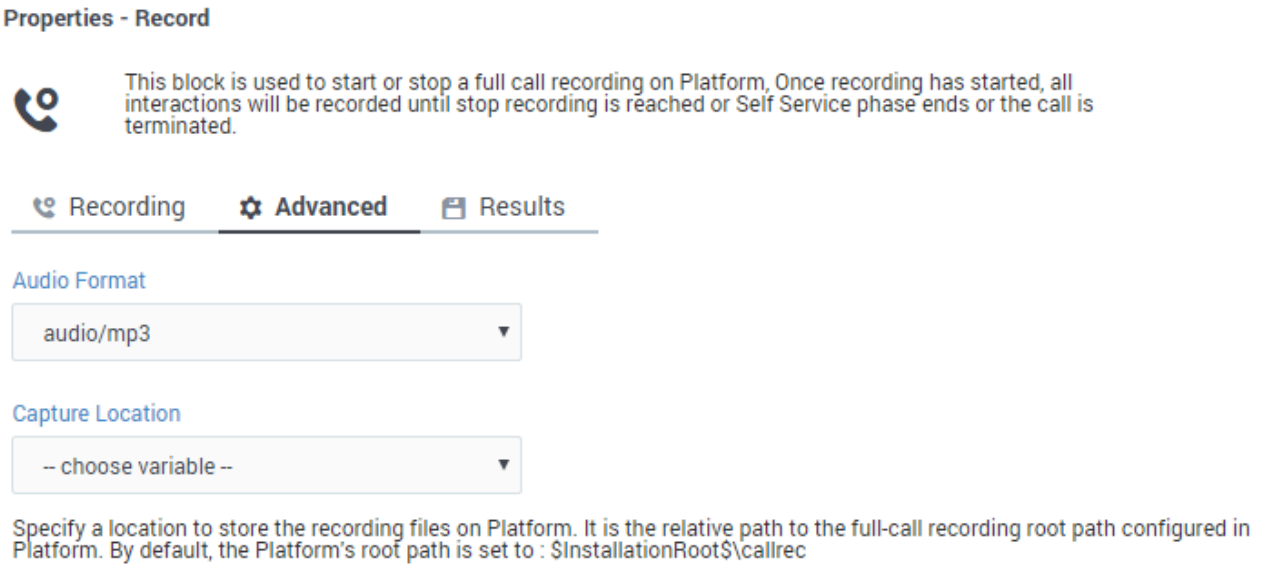
Advanced tab

If you selected a variable to control recording on the **Recording** tab, the **Advanced** tab is available.

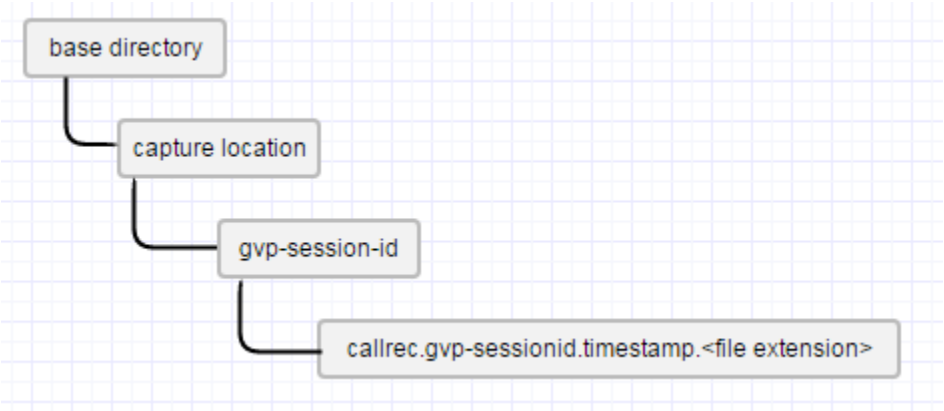
The options shown on this tab are based on whether your site is using a FCR or GIR recording type. Scroll down to the section that applies to your installation.

FCR

Select an **Audio Format** (the default audio format is **mp3**) and a **Capture Location**.



The audio file will be stored on the platform, in a location relative to the base directory (typically, this is `$InstallationRoot$\callrec`). This diagram shows the directory and file-naming structure:



> **Important**
> If the **Capture Location** is not specified, this sub-folder is not created.

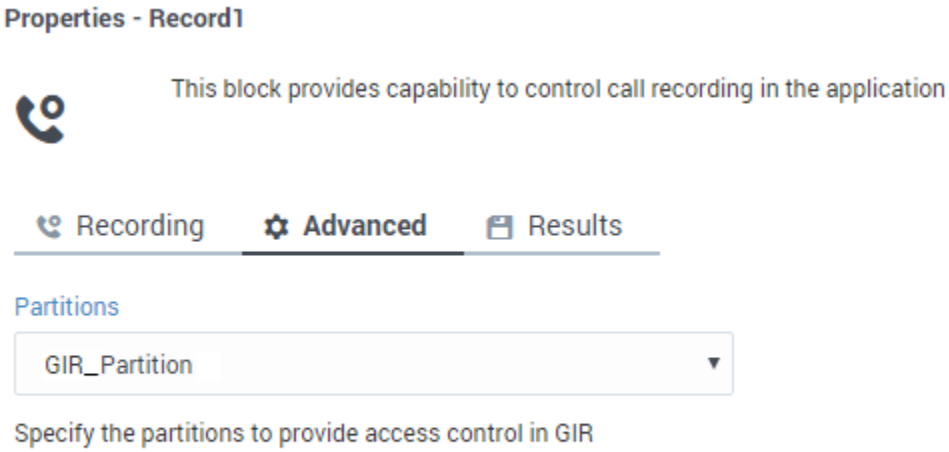
GIR

GIR uses *partitions* for controlling access to media files, such as call recordings.

Designer applications will tag each recording with the partitions specified in this block so that GIR receives both the recordings and their associated partitions. GIR will then use them to control access to recordings. Only those users who belong to the partition will have access to the recordings tagged with that partition.

The partition for the overall application is set by the *DefaultPartition* system variable during the initialization phase. However, if you want this block to use a different partition for GIR access control, you can select one from the **Partitions** list.

This will override the default partition setting, but only for this block.



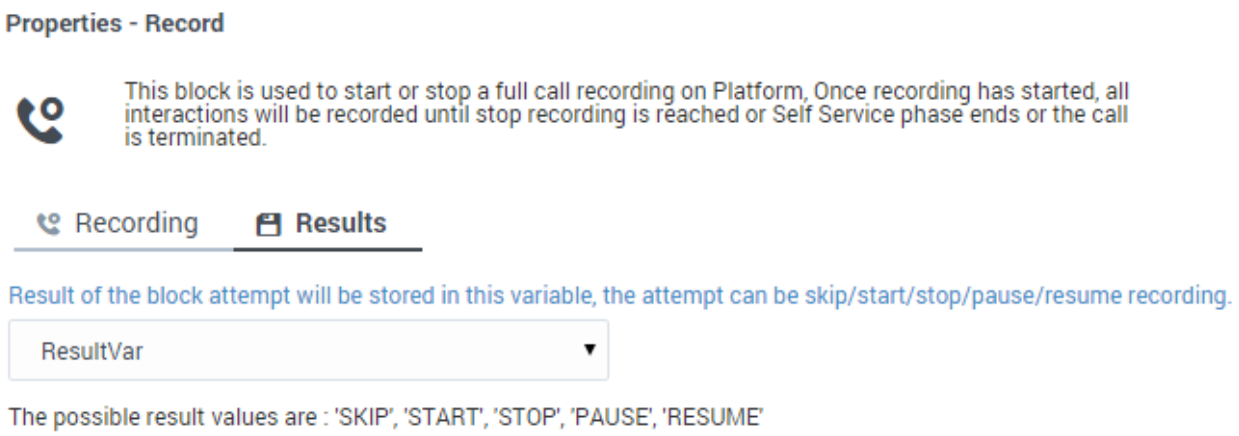
Results tab

Select a variable that will store the result of the block attempt.

If you are using FCR as a call recording type, the result indicates whether the block has attempted to **start**, **stop**, or **skip** recording.

If you are using GIR as a call recording type, the result can also indicate an attempt to **pause** or

resume recording.



Updating applications to use a new recording type

The **Record** block can use either FCR or GIR for call recordings. GIR offers more capabilities, such as **pause** and **resume**, that are not offered by FCR.

If the recording type is changed in your environment (this change can only be done by Genesys), the previous recording type will continue to be in effect until you publish your applications. You might also need to adjust your applications to the new set of capabilities, as follows:

Moving from FCR to GIR

The existing FCR-based **Record** blocks will continue to work with GIR without changes. To use the additional capabilities offered by GIR, change your blocks to select the new options.

When you are finished making changes, publish the updated applications.

Moving from GIR to FCR

FCR offers fewer options, so you will no longer have access to **pause** or **resume** recording options. Therefore, **Record** blocks that specify these actions will no longer work and must be changed to use either **start** or **stop**.

When you are finished making changes, publish the updated applications.

> **Important**
>
> The **Advanced** tab will only show those options that are applicable to the current recording type.

Record Utterance block

Use this block to capture a voice recording of the caller. You can then use the HTTP REST block to send the recording to an external API, or play it back using the Play Message block.

This block can only be used in the **Self Service** phase of an application. After the Self Service phase is completed, the recording is no longer available.

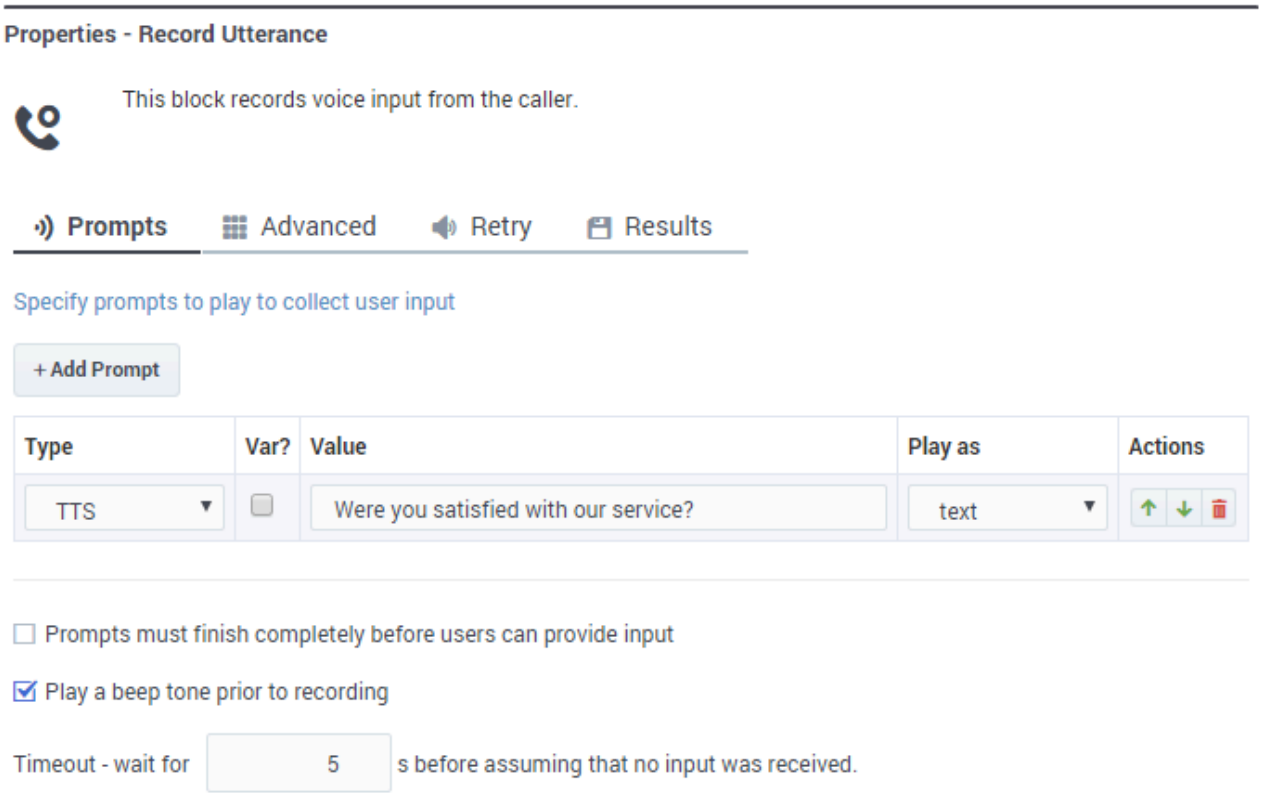
Prompts tab

Click **Add Prompt** to specify the prompts that will be played to the caller.

Select **Prompts must finish completely before users can provide input** to prevent users from responding to the prompt before it has finished.

Select **Play a beep tone prior to recording** to indicate that recording is about to begin.

You can also specify a timeout value to indicate how long Designer should wait for the user to provide a voice input before moving to the next block.

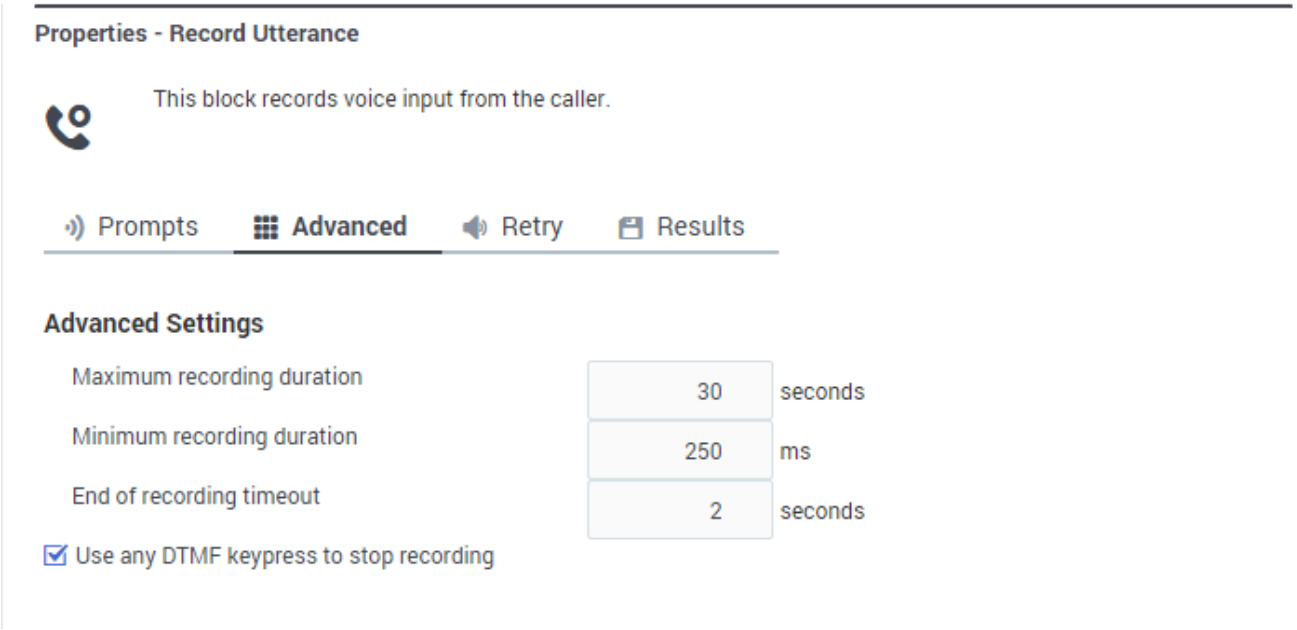


Advanced tab

Specify how many seconds of recording to capture by setting the minimum and maximum duration values. The default duration is 10 seconds.

You can also specify a timeout value to indicate many seconds of silence should pass before recording stops. The default value is 2 seconds.

Select **Use any DTMF keypress to stop recording** to allow users to stop the the recording with any key press.



Retry tab

Enable **Use application-wide retry** to use the default retry settings specified in your application settings.

If you disable **Use application-wide retry**, you can enable **Allow Retries** to use the standard input retries if no input was detected during recording.

Results tab

Specify the variables that will store the recording and its details.

User Input Block

You can use the **User Input** block in the **Self Service** phase to collect information from the user, such as an account number or credit-card information, and store it in a variable for processing. In the **Assisted Service** phase, you can use this block to gather more information from the user.

Optionally, you can specify whether retries are allowed if the input is not recognized, and whether to play a retry message, along with the original prompt message.

> **Tip**
>
> If the user enters invalid information or no input, then the value of the results variable is undefined. This must be considered before any later block can process the result. For example, a **Segmentation** block could determine whether the results variable stores a valid value and, based on the result, branches to different paths.

Prompts tab

Disable barge-in

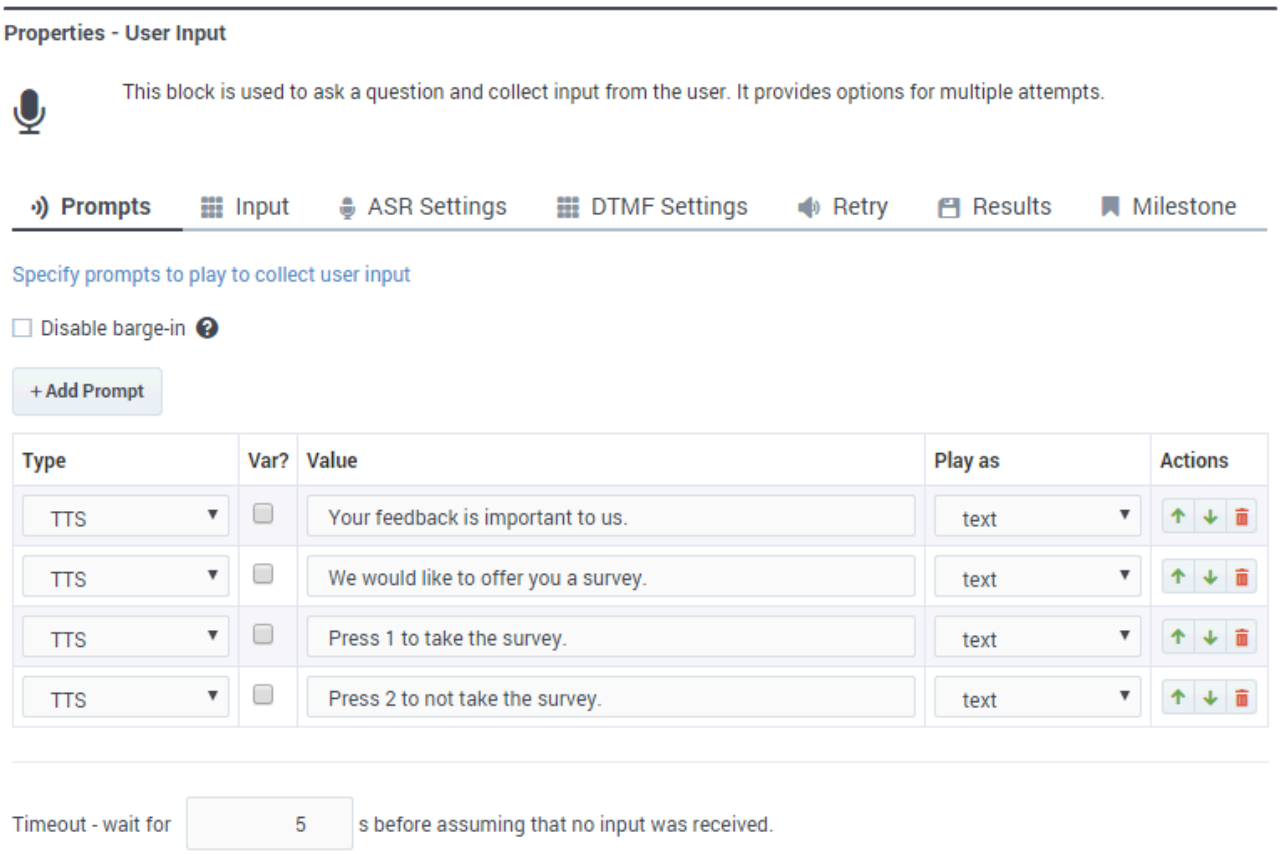
Select this option to prevent callers from interrupting a prompt while it is still playing. For example, you might want a "Welcome" message to play all the way through before the caller can enter another command and skip to the next menu prompt.

If this option is not selected, barge-in is enabled, and the prompt can be interrupted by the caller.

> **Important**
>
> The selected barge-in setting applies irrespective of whether global DTMF commands are used or not.

Click **Add Prompt** to play prompts when the menu starts.

Set the timeout period, in seconds, to wait before assuming that no input was received from the caller. Refer to the **Retry** tab to specify which actions are taken if the timeout period is reached. If retries are not permitted and the timeout period is reached, the application moves onto the next block.



Input Tab

Choose one of the following options:

Built-in Grammar

Select this option to use a built-in grammar. You can select from the following types:

- boolean
- currency
- date
- digits
- number
- phone
- time

If you select **digits**, you can also set the following options:

- **Minimum number of input digits** — Specify the minimum number of digits that the caller must enter.
- **Maximum number of input digits** — Specify the maximum number of digits that the caller can enter.

Next, specify the input mode for the grammar. You can select **DTMF**, **Speech**, or both. (If you select only **Speech** mode, **DTMF** grammars remain active but are not matched.)

Languages for built-in grammars can be managed using the *AppLanguageName* system variable (see System variables) or the Change Language block.

External Grammars

Select this option if you have created your own speech grammar. Next, click **Add Grammar** to add one or more speech grammars to use with this block. Provide the following information:

- **Var?** - Enable this check box to indicate that the selected speech grammar will be determined by a variable.
- **Dynamic?** - Enable this check box to indicate that the selected speech grammar contains dynamic values that can change over time (for example, an employee directory). For more information, see Dynamic Grammars.
- **Name** - Select the speech grammar name (or variable) that you want to add to this block.
- **Mode** - Specifies whether this speech grammar is for voice or DTMF.

ASR Settings tab

Use application-wide ASR settings

Enable this check box to use the default ASR (Automatic Speech Recognition) settings for your application. You can view or change these settings by clicking **Settings** in the Toolbar.

If you disable the **Use application-wide ASR settings** check box, you can set the following options for this block:

- **Confidence Level** - Specifies the speech recognition confidence level. If the caller's input is below this threshold, the input is determined as **No Match**. A value of `0.0` specifies that minimum confidence is needed for a match. A value of `1.0` specifies that maximum confidence is required before a match is determined.
- **Sensitivity** - Specifies the sensitivity level. A value of `1.0` specifies that speech recognition is highly sensitive to quiet input. A value of `0.0` specifies that speech recognition is least sensitive to noise.
- **Speed vs. Accuracy** - Specifies the balance between how fast the application responds to the input versus how accurate the response is interpreted. A value of `0.0` specifies that quick recognition is preferred. A value of `1.0` specifies that high accuracy is preferred.
- **Complete Timeout** - Specifies the required length of silence, in seconds, following user speech before the application determines a result (match, **No Match**, or **No Input**).
- **Incomplete Timeout** - Specifies the required length of silence, in seconds, following user speech before the application determines a result. This property is used in the following situations:

  - If the speech prior to the silence does not match all active grammars, this property specifies how long to wait before the partial result is rejected as **No Match**.
  - If the speech prior to the silence matches an active speech grammar, but it is still permissible to continue speaking and match the speech grammar. By contrast, **Complete Timeout** applies when the speech prior to the silence matches an active speech grammar and no further words are permissible.
- **Max Speech Timeout** - Specifies the maximum amount of time, in seconds, for which speech input is allowed before it is determined to be **No Match**.

> **Tip**
> If you change a setting and you later want to revert the setting to the default value, click **Global**.

DTMF Settings tab

Configure the following settings for DTMF input:

- **Input termination character** - Specify a termination character that the caller can enter to mark the end of the input string. Commonly, * or # are used as termination characters. If the caller does not enter a termination character, the application waits until the **Terminating Timeout** period has passed before processing the input.
  Example: You set this value to #. The caller enters 1234#. The input is **1234** and **#** signals that no more characters will be entered.
- **Interdigit Timeout** is the amount of time, in seconds, that the application waits between digit inputs before assuming the end of the input string. If the user entered too few or too many digits, a retry is attempted. If retries are not allowed, the application moves on to the next block.
  Example: You set this value to 3 and specify that the input can be between three and five digits. The caller enters 1234. The application waits **3** seconds before assuming a fifth digit will not be entered.
- **Terminating Timeout** is the amount of time, in seconds, that the application waits for the **Input Termination Character** before processing the input string. If the input is always a static length (for example, four characters), then you can set this value to `0` for the application to immediately process the input after the last digit is entered.
  Example: If this value is **5** and the caller enters 1234, the application waits **5** seconds before processing the input.

Retry tab

Use application-wide retry

Enable this option to use the default retry settings for your application. You can view or change these settings by clicking **Settings** in the Toolbar.

Allow retries

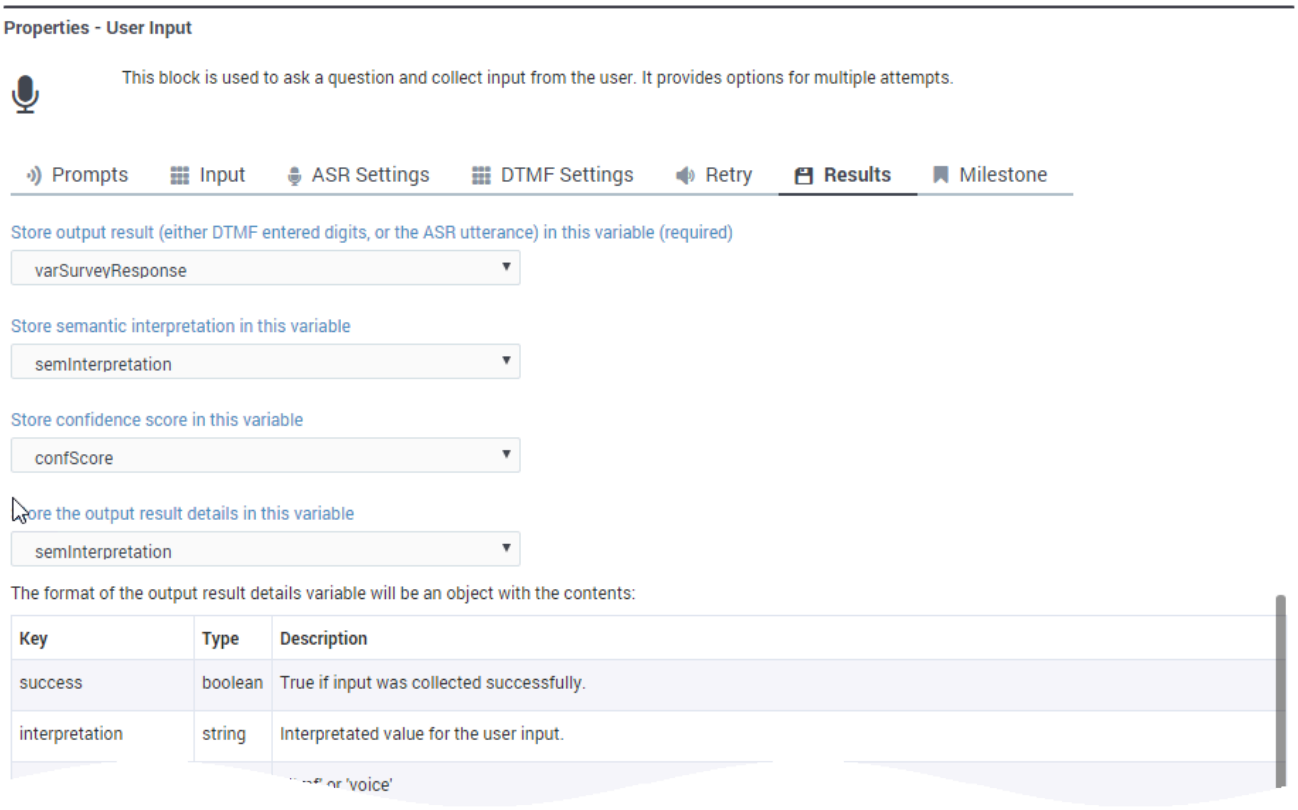
If you disable the **Use application-wide retry** check box, you can enable **Allow retries** to specify retry rules for this block. You can set the following options:

- **Number of No Input retries allowed**
  Select the number of retries to allow for callers whom do not provide input. For each retry, you can specify whether a prompt is played by clicking the corresponding section beneath this field. For example, if you allow two no-input retries and you want to play a prompt after the first retry, select the **No Input #1** line and add a prompt. Enable the **Play original menu prompt after this retry prompt** check box to repeat the menu prompts for the caller.
- **Number of No Match retries allowed**
  Select the number of retries to allow for callers whom do not provide a match for a **Menu Block**. For each retry, you can specify whether a prompt is played by clicking the corresponding section beneath this field. For example, if you allow two no-input retries and you want to play a prompt after the first retry, select the **No Match #1** line and add a prompt. Enable the **Play original menu prompt after this retry prompt** check box to repeat the menu prompts for the caller.
- **After Final No Input**
  Add the prompt to play after the maximum number of permitted No Input retries is reached. If this block is in the Self Service phase, you can also specify a target destination for the application to jump to, such as another block in the Self Service phase or to the Assisted Service or Finalize phase of the application.
- **After Final No Match**
  Add the prompt to play after the maximum number of permitted No Match retries is reached. If this block is in the Self Service phase, you can also specify a target destination for the application to jump to, such as another block in the Self Service phase or to the Assisted Service or Finalize phase of the application.

Results tab

Specify the variables in which to store the results of the interaction, semantic interpretation, confidence score, and output result for speech recognition (such as the **Confidence Level** value).

Example



Milestone tab

Specify a milestone for this block.

Example scenario

If you want to:

- Collect a single digit and allow for two retries. The caller can start collecting input only after the entire audio prompt has finished playing.
  - **Disable barge-in:** Enabled
  - **Minimum number of input digits**: 1
  - **Maximum number of input digits**: 1
  - **Allow retries**: Enabled
  - **Number of No Input retries allowed**: 2
  - **Number of No Match retries allowed**: 2

Dynamic Grammars

A dynamic grammar contains an array of values that automatically update in response to external input. Whereas traditional grammars are static and must be updated manually for each change, dynamic grammars are always current and do not require manual updates. This is useful for situations in which the grammar contents change frequently, such as employee or customer lists.

Consider the following example. You have created a voice application with a **User Input** block that allows the caller to contact specific employees in your company. You are using an external grammar that contains the names of all of your employees as valid inputs for this block. However, if an employee leaves the company or your company hires a new employee, the grammar is no longer current and must be updated. A dynamic grammar, however, can use an array of values from a variable and update itself based on external input. You do not need to manually update a dynamic grammar each time there is a change to the list.

There are two ways to create an array for use with dynamic grammars:

- Use an HTTP REST block.
- Use an Assign Variables block.

Using an **HTTP REST** Block

This method uses an **HTTP REST** block to fetch an array from a web service.

In this example, we are using a web service that returns the following JSON data:

```
{
        "success": true,
        "employees": ["John", "Julie", "Mark"]
}
```

Next, in the **Output Parameters** section of the **HTTP REST** block, we can assign **employees** to a User Variable that we previously created, called **varEmployees**.

Finally, in the **User Input** block, we can select **varEmployees** as a dynamic grammar. Each time the application runs, the **HTTP REST** block fetches the employee list and uses its contents as a dynamic grammar for the **User Input** block.

Using an **Assign Variables** Block

This method uses an **Assign Variables** block to push values to an array.

In this example, we have initialized a variable called **varEmployees** with a value of `'[]'` (an empty array).

Next, in the **Assign Variables** block, we use the expression `varEmployees.push('John')` to add an employee, **John**, to the employee list.

You can use multiple **Assign Variables** blocks to add items to the array.

Chat Message Block

You can use the **Chat Message** block to send a chat message to a contact. You can create a custom plain text message, or use one of the standard responses.

This block can also be used as a busy treatment.

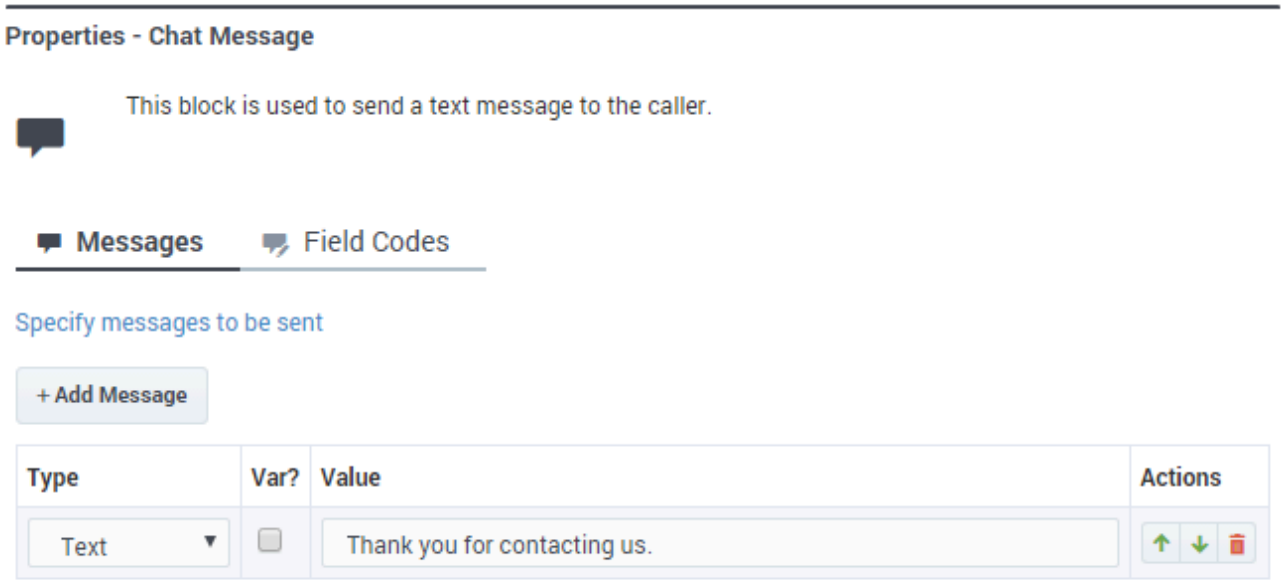
Messages tab

Use the **Messages** tab to add and manage chat messages.

Select **Text** if you are writing a custom text message. Enter the message in the **Value** field.

If you want to specify the value with a variable, select **Var** and choose the appropriate variable.

Select **Message** if you want to use a standard response. Click the "picker" icon to open the Chat Resource Set and select the message you want to use.



Field Codes tab

(Optional) If you are using standard responses, you can use the **Field Codes** tab to specify the field codes being used.

Chat Custom Message Block

You can use the **Chat Custom Message** block to send a custom chat message to a contact. You can create a message using plain text, or specify a variable that contains the message you want to send.

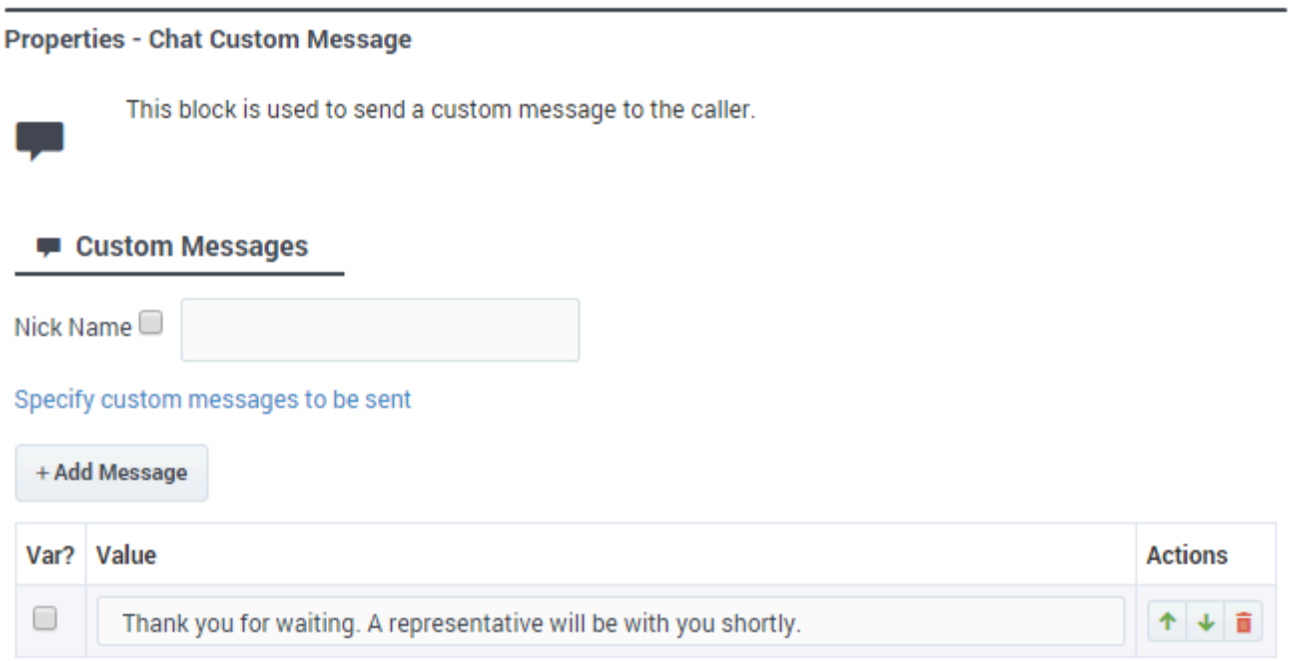
> **Important**
>
> The **Chat Custom Message** block does not support Standard Responses or Field Codes. It also can't be used as a busy treatment.

Custom Messages tab

Use the **Custom Messages** tab to add and manage chat messages.

Nick Name (optional) is the name that chat contacts will see as being the sender of the custom chat message. You can enter a name, select a variable, or leave this field blank.

Click **Add Message** to enter the value of the message. If you want to specify the value with a variable, select **Var** and choose the appropriate variable.



Chat Transcript Block

The **Chat Transcript** block lets you send a transcript of the chat to the email address specified in a contact's profile.

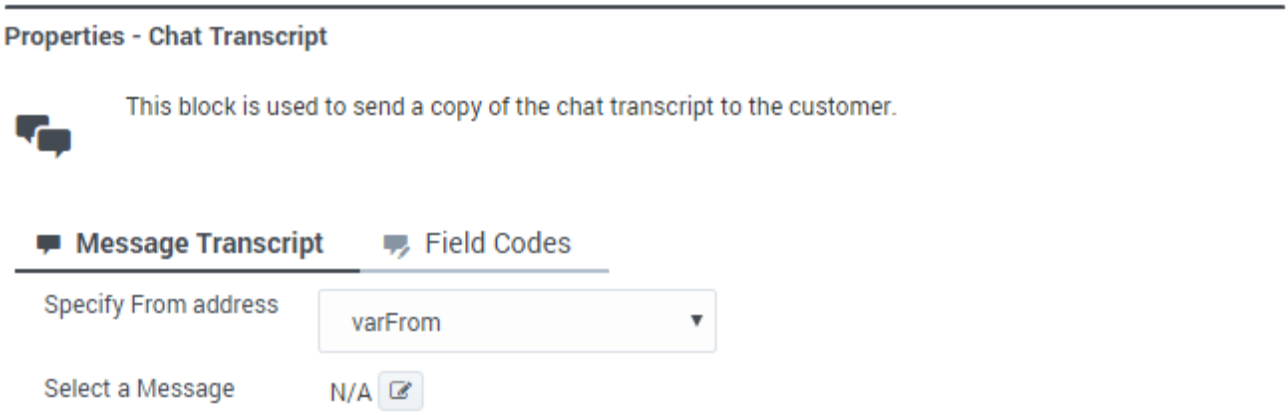
When used within the initial application flow, the transcript message is sent right away. Note that if routing is not yet completed, this transcript will not include any messages or conversations that take place between the agent and the contact after that point.

If you want all messages or conversations that take place between the agent and the contact to be included in the transcript, add this block to a post-processing application that you have specified in the **Advanced** tab of the Route block.

Message Transcript tab

Use the **Message Transcript** tab to select the variable that contains the email address you want to use as the *From* address in the emailed transcript.

For **Select a Message**, click the "picker" icon to open the Chat Resource Set and select a standard response message to include with the transcript.



Field Codes tab

(Optional) If you are using standard responses, you can use the **Field Codes** tab to specify the field codes being used.

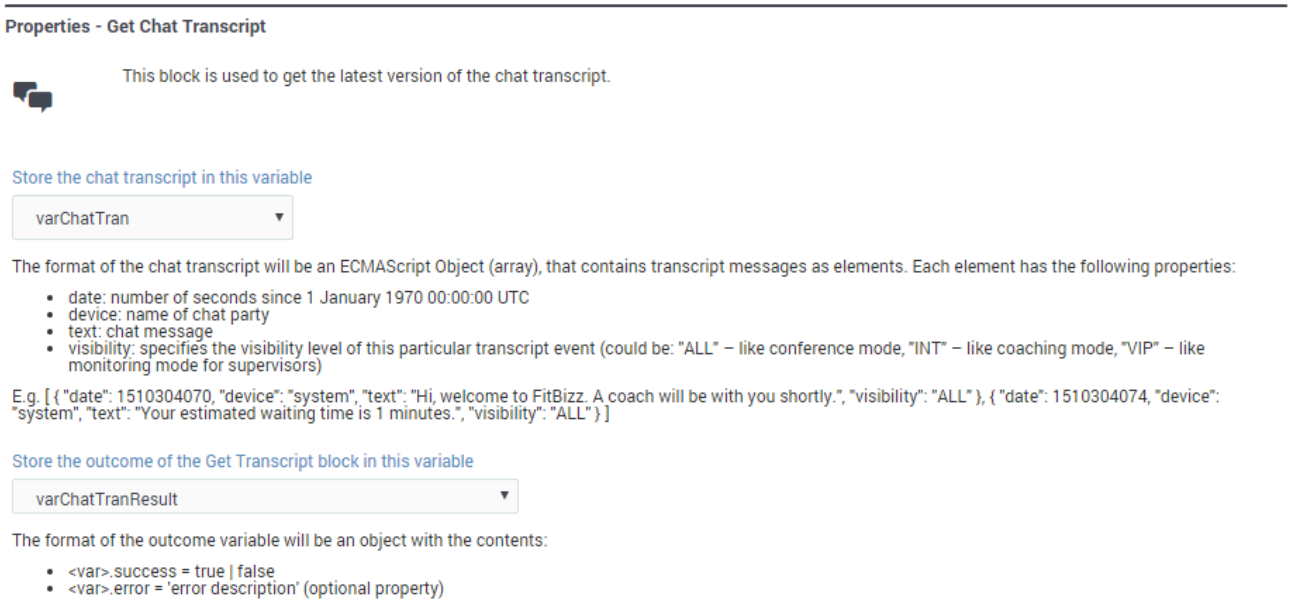
Get Chat Transcript Block

The **Get Chat Transcript** block enables you to store the contents of the latest chat transcript in a variable which can then be referenced at a later point in the application flow. For example, you might want to retrieve the chat transcript and send it to multiple email recipients.

Using this block

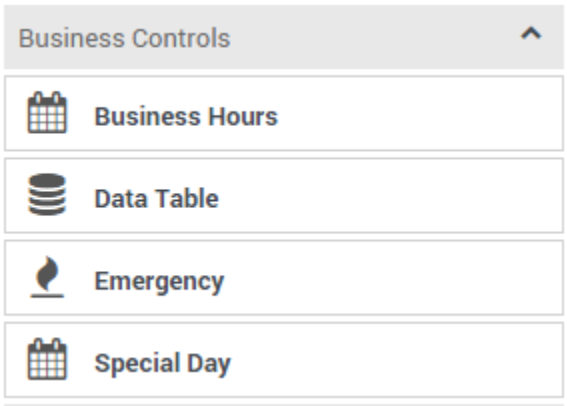
This block can be used in the **Assisted Service** and **Finalize** phases.

Select a variable to store the chat transcript and a variable to store the result of the **Get Chat Transcript** request.



Business Controls Blocks

The blocks in this category are used to control various operational aspects of your business, such as setting up your hours of operation, emergency flags, data tables, special days, and so on.



Business Hours

Announce when your business is closed.

Used in: **Initialize**, **Self Service**, **Assisted Service**

Data Tables

Read values from a data table.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Emergency

Add a conditional emergency option to your application.

Used in: **Initialize**, **Self Service**, **Assisted Service**

Special Day

Define holidays and other special days.

Used in: **Initialize**, **Self Service**, **Assisted Service**

Business Hours Block

You can use the **Business Hours** block in the **Initialize**, **Self Service**, or **Assisted Service** phase to announce when your business is closed. You can also choose to end the call at this point.

> **Tip**
>
> The hours that you define in the **Business Hours** block are based on the application's time zone setting. To set the application time zone, select the **Initialize** phase and open the **System Variables** tab. Click the drop-down menu in the **Timezone** row and select a value. You can override the default time zone setting by selecting a variable in the **Set Timezone** section of the **Business Hours** tab.

Business Hours tab

To set your business hours, select each **Day** you are open and specify the **Start Time** and **End Time** for each day.

Open All Day

Select if your business is open for that entire day.

No End Time

Select if there is no set end time for the given day (this option only appears if the business is not closed for that day and the **Open All Day** option is not enabled).

Terminate the call if it is outside Business Hours

Select to end calls that come in outside of business hours.

Use Business Hours defined in Business Controls

If you prefer to use a specific shared business hours resource that you've defined on the Business Controls page, enable this option and select it from the list of defined business hours resources.

Alternatively, you can specify a variable to be used dynamically while the application runs. Select **Variable?** to specify a user-defined variable that holds the name of a Business Hours resource. If this resource will be read from a data table, you must also select **From Data Table?** to indicate that the variable holds the result of a data table lookup. Otherwise, the result will not be evaluated correctly.

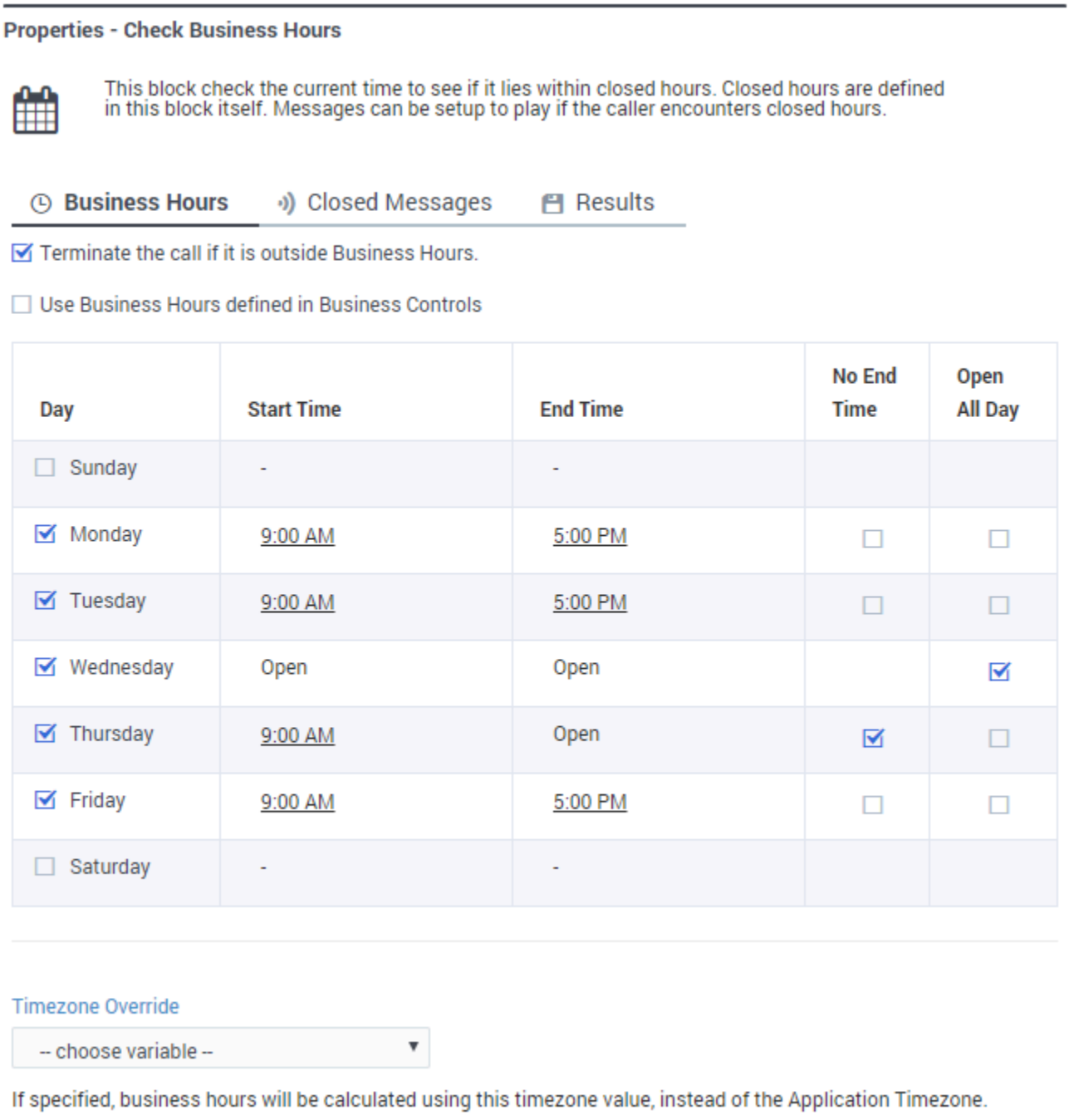
Important

- If the Business Hours are being determined dynamically at runtime, you can't mix user-defined variables (for example, `varDepartmentName + “_PrimaryHours"`) with variables retrieved from data table lookups. Make sure you check the appropriate box to indicate the type of variable being used.
- No matter which method you use, the name stored in the variable must match one of the Business Hour objects you created on the Business Controls page.

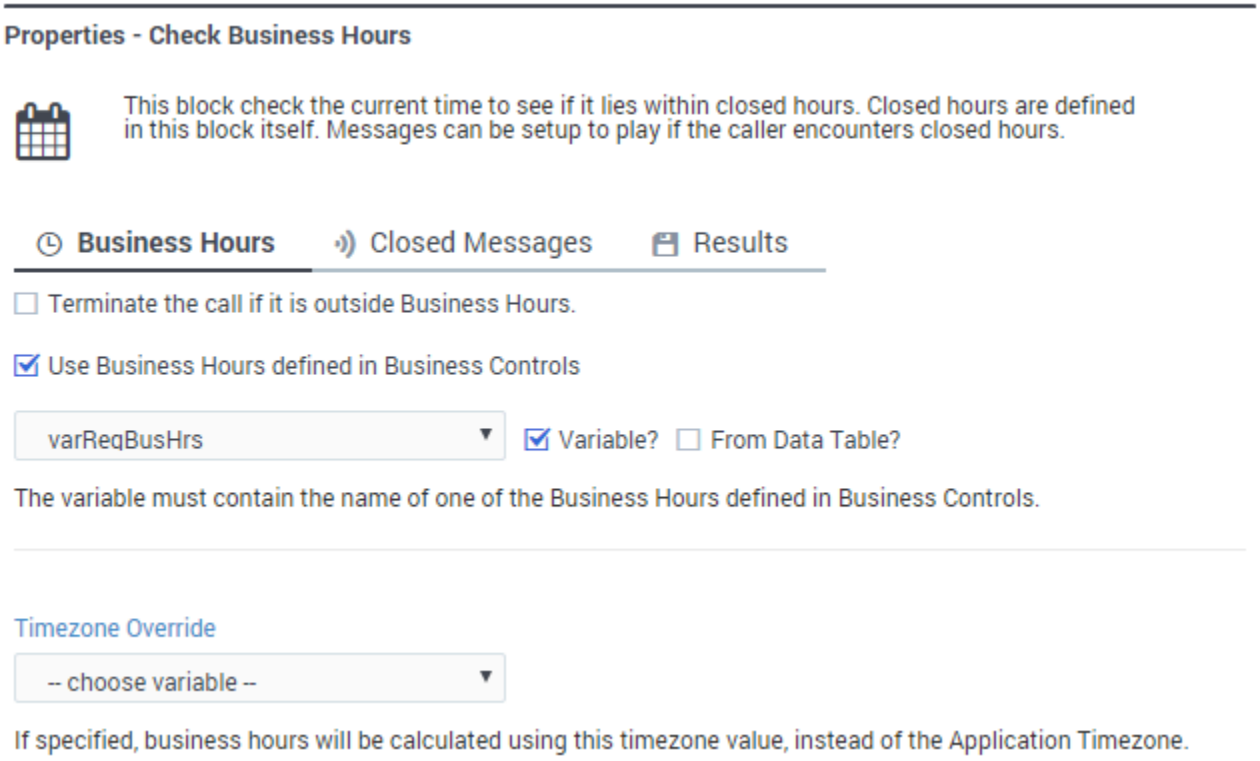
Timezone Override

Select a variable that will override the time zone setting for the application.

Example 1


Properties - Check Business Hours
This block check the current time to see if it lies within closed hours. Closed hours are defined in this block itself. Messages can be setup to play if the caller encounters closed hours.
Business Hours
Closed Messages
Results
Terminate the call if it is outside Business Hours.
Use Business Hours defined in Business Controls
Day
Start Time
End Time
No End Time
Open All Day
Sunday - -
Monday 9:00 AM 5:00 PM
Tuesday 9:00 AM 5:00 PM
Wednesday Open Open
Thursday 9:00 AM Open
Friday 9:00 AM 5:00 PM
Saturday - -
Timezone Override
-- choose variable --
If specified, business hours will be calculated using this timezone value, instead of the Application Timezone.

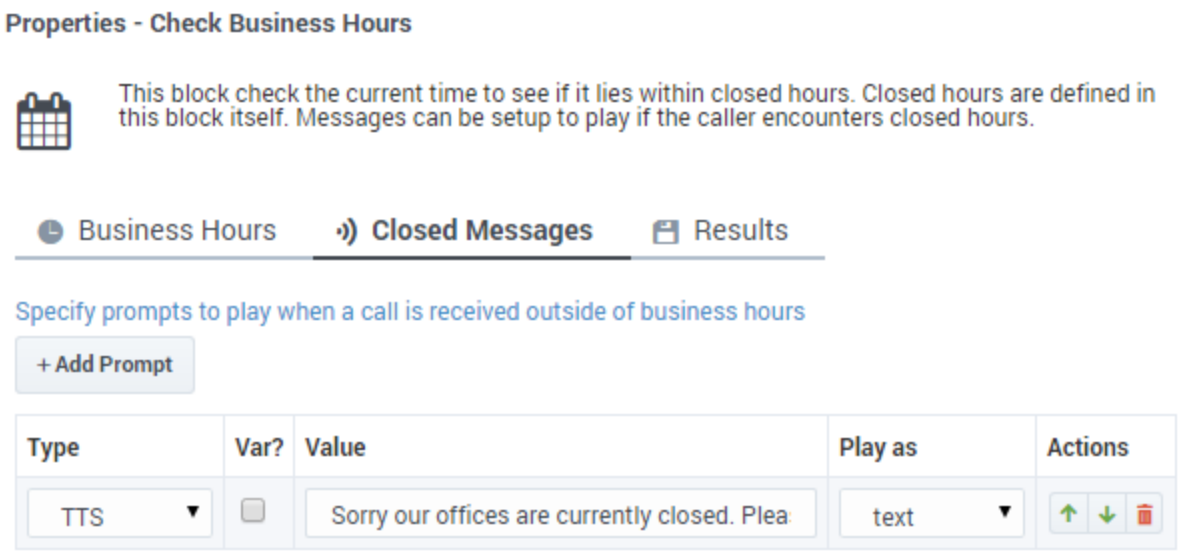
Example 2



Closed Messages tab

In the **Closed Messages** tab, you can select messages to play to callers when your business is closed. Click **Add Prompt** and complete the form to create a new message.

> **Tip**
> See the Play Message block page for information on how to use prompts.



Results tab

You can assign a variable to the **Set the result of Block operation status in this variable** property to store the result of the block operation check. If specified, the variable is assigned the Boolean value of **true** or **false**, to indicate if the block operation completed without errors.

You can assign a variable to the **Store the result of Business hours check in this variable** property if you need to use the result of the check later in application. If specified, the variable is assigned the Boolean value of **true** or **false**.

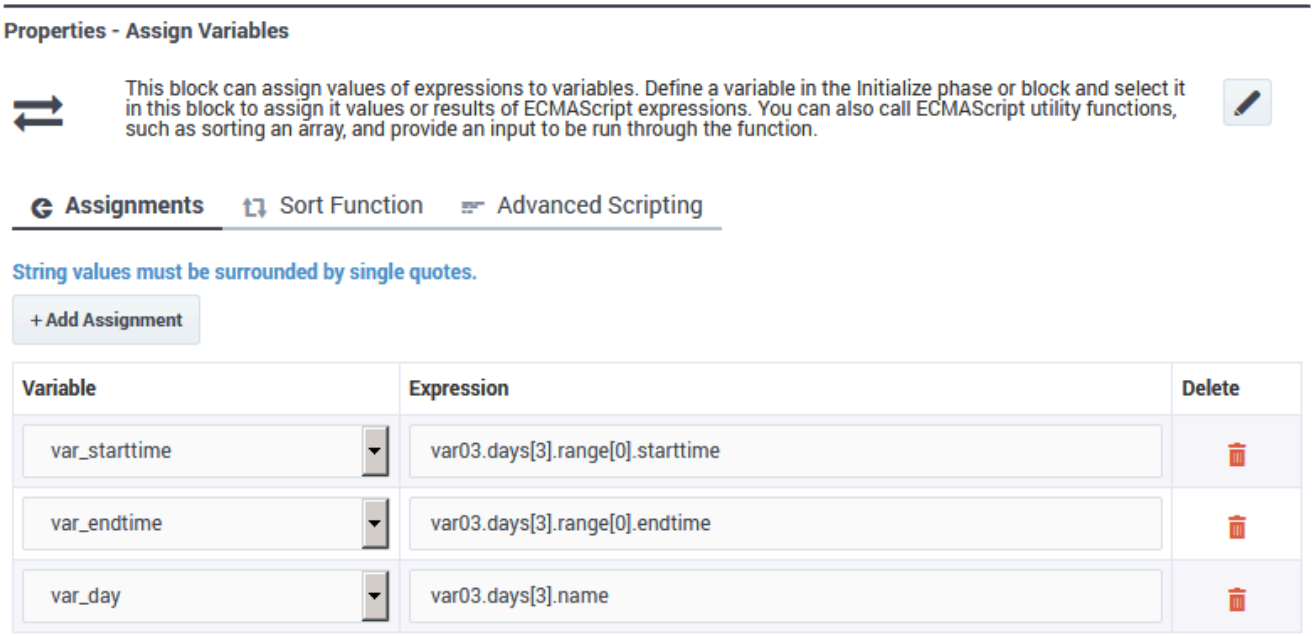
You can assign a variable to the **Store the business hours schedule in this variable** property if you need to read the business hours schedule later in the application. If specified, the variable is assigned a JSON object. The JSON object structure is:

```
{
"hours": [
{ "day": "Sunday", "closed": true, "starttime": "0900", "endtime": "1700" }
,
{ "day": "Monday", "starttime": "0900", "endtime": "1700" }
,
{ "day": "Tuesday", "starttime": "0900", "endtime": "1700" }
,
{ "day": "Wednesday", "starttime": "0900", "endtime": "1700" }
,
{ "day": "Thursday", "starttime": "0900", "endtime": "1700" }
,
{ "day": "Friday", "starttime": "0800", "endtime": "0900" }
,
{ "day": "Saturday", "starttime": "1000", "endtime": "1600" }
]
}
```

This JSON object can be used in expressions for the **Assign Variables** block and prompt values for the **Play Message** block.

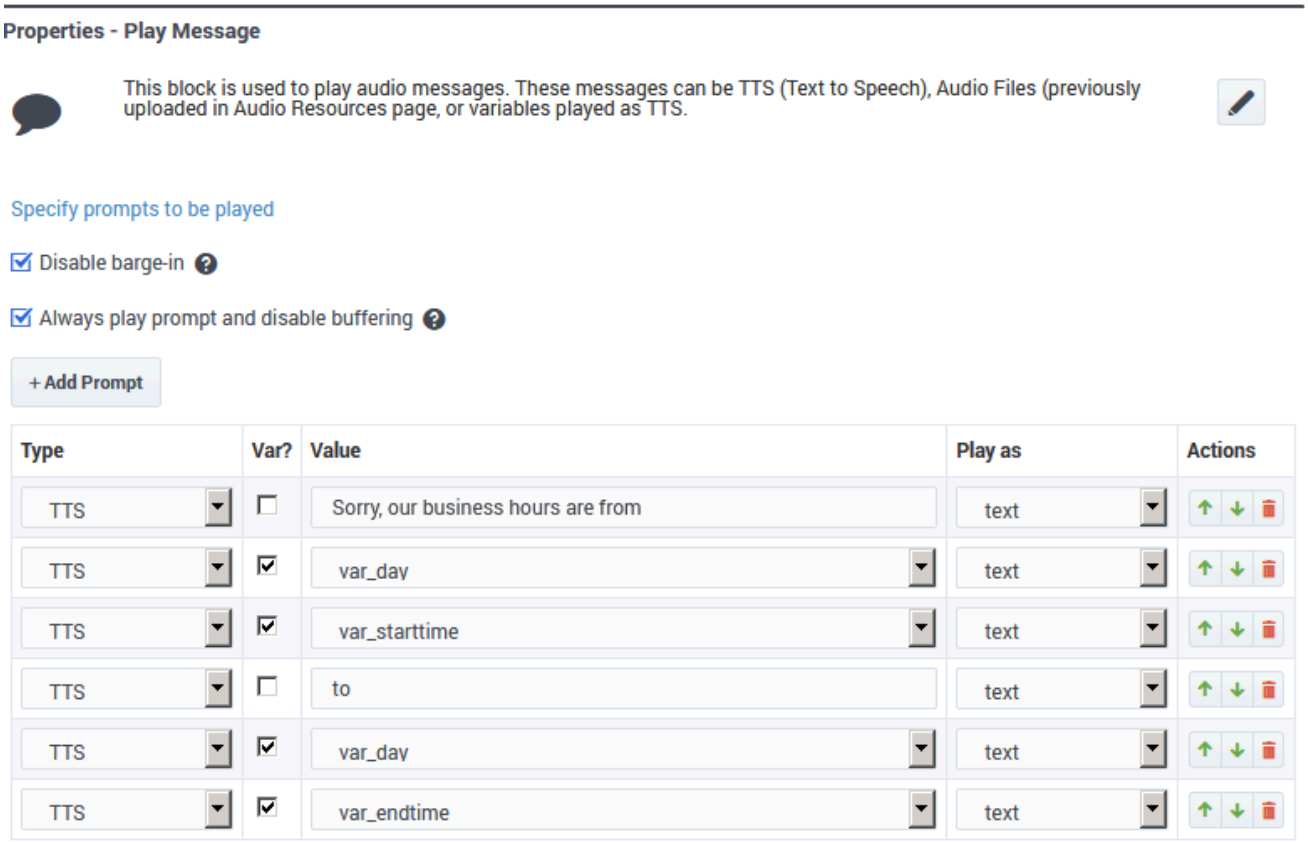
Using Business Hour Values in Prompts

You can also use the **Business Hours** block to announce business hours in prompts. For example, you could use an Assign Variables block to assign the following variables:



Then, set up the prompt to announce the **End Time** value for Wednesday:

- In the **Play Message** block, click **Add Prompt**.
- Choose **TTS** as the **Type**.
- Enable the **Var?** checkbox, and select **var_endtime** as the **Value**.



> **Important**
>
> The `days[n]` range is from from `0` to `6`, with `0` representing Sunday and `6` representing Saturday.

Scenarios

If you want to:

- Specify that the business is closed after 4 p.m. on Thursdays
  - Set the **End Time** value on Thursday to 4:00 PM
- Play a message if a customer calls after business hours, and then end the call
  - On the **Business Hours** tab, enable the **Terminate the call if it is outside Business Hours** checkbox.
  - On the **Closed Messages** tab, create a prompt.

- Specify business hours for a different timezone than the one the application is running in
  - In the **Intialize** phase, assign a value to the **timezone** variable (remember that string values must be surrounded by single quotes — for example, 'UTC').
  - On the **Business Hours** tab, select the **timezone** variable from the **Set Timezone** section.

Data Tables Block

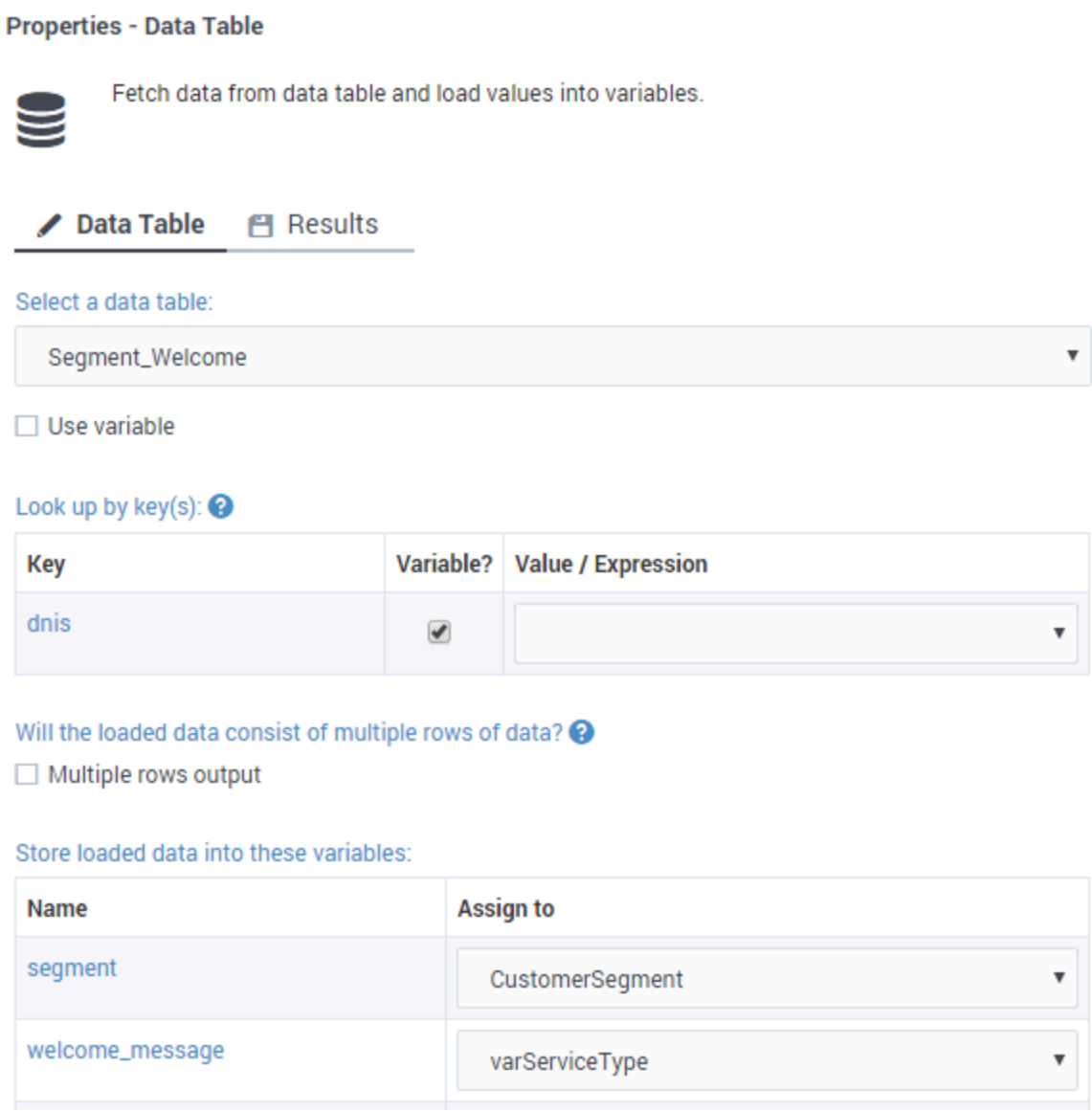
You can use the **Data Tables** block in any phase of your application to read values from a Data Table.

Data Table tab

Select a Data Table in the drop-down list. If you want to enable the option to use the data table as a variable, select **Use variable**. For example, you might select this option if you are using the same application in multiple locations, and each location needs to refer to a data table that is specific to that location.

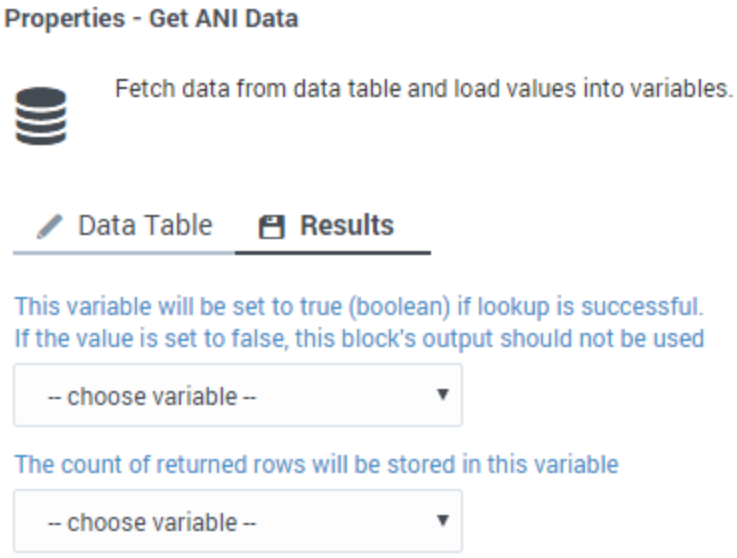
Once you have selected a Data Table, you must configure the following:

- **Look up by key(s)** - For each key in your Data Table, enter a value (or variable, if **Variable?** is enabled) to use as an input for the **Data Table** block.
- **Multiple rows output** - Select this option if the lookup key is of a date/time range data type, or if the data table contains multiple keys.
- **Store loaded data into these variables** - For each column in your Data Table, select a variable to hold the output value of your **Data Table** block.



Results tab

Select a variable to store the outcome status of the lookup (**true** or **false**). You can also select a variable to store the number of returned rows.


Properties - Get ANI Data
Fetch data from data table and load values into variables.
Data Table
Results
This variable will be set to true (boolean) if lookup is successful.
If the value is set to false, this block's output should not be used
-- choose variable --
The count of returned rows will be stored in this variable
-- choose variable --

Emergency Block

You can use the **Emergency** block in the **Initialize**, **Self Service**, or **Assisted Service** phase to implement a conditional emergency option in your application.

You can configure this block to play an emergency message and then optionally terminate the call. This process works only if the emergency mode switch is set to **ON** in either the **Emergency** block or in the Emergency Flags section.

If the switch is set to **OFF**, the block has no effect and it is skipped by the application.

For simple applications, a user typically places this block at the start of the **Self Service** phase. If service is disrupted, the **Emergency** block is easy to locate and enable.

For complex applications that branch into multiple geographic areas, you can place this block in certain segments of a Segmentation block that uses logic to detect branches that are affected by emergency conditions. This allows selective enabling of emergency mode for calls that require services from affected branches. For example, if your company has two offices and one is closed due to an emergency, you can route calls to the other office.

> **Tip**
> Remember to set the emergency mode switch to **OFF** once normal operation resumes.

Using this Block

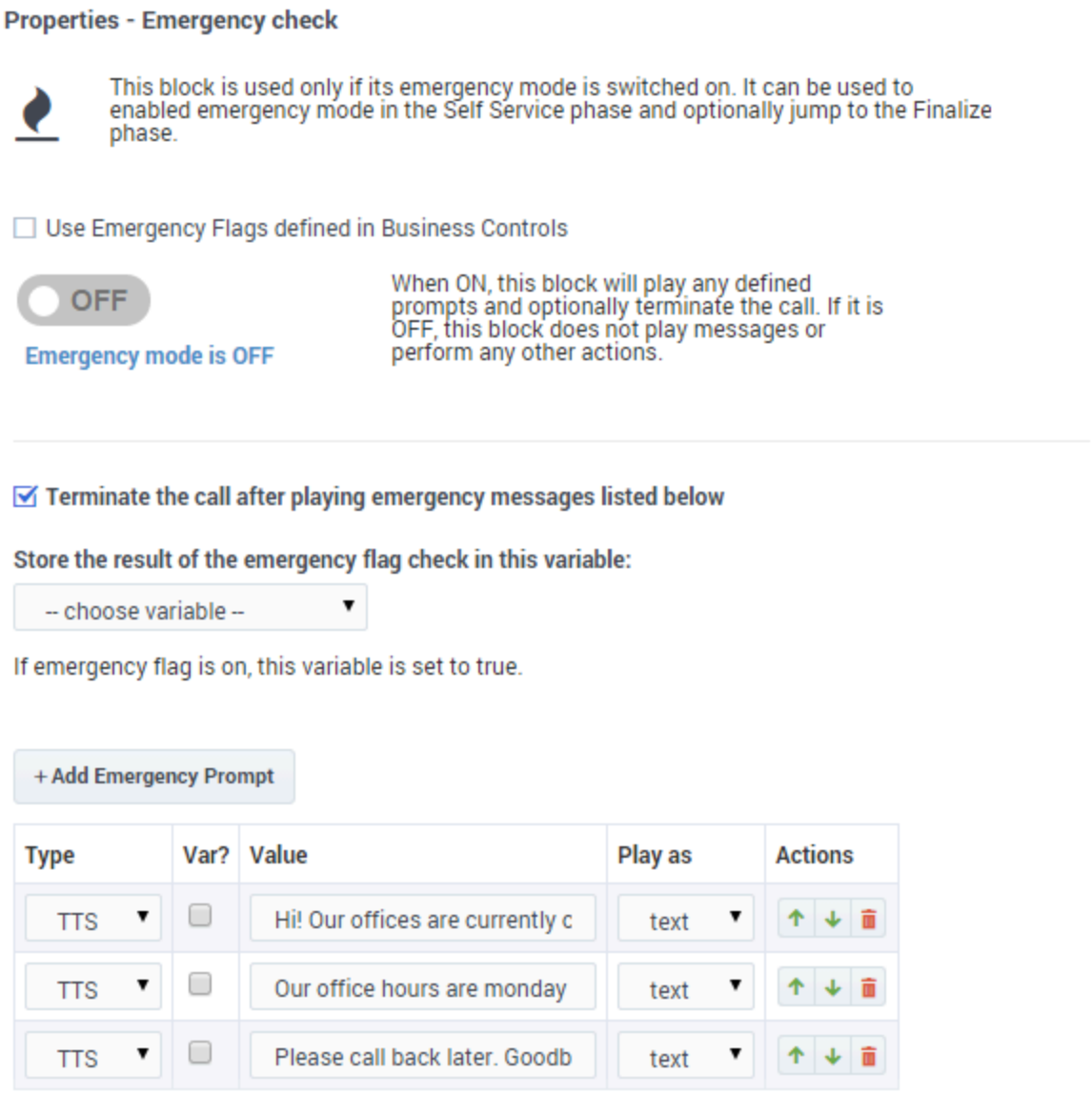
If you have defined an Emergency Flag, enable the **Use Emergency Flags defined in Business Controls** check box. Otherwise, follow the instructions below.

To start, ensure the emergency mode switch is set to **OFF**. You can set this switch to **ON** in an emergency situation.

Click **Add Emergency Prompt** to add one or more emergency prompts to play to callers when the emergency mode switch is set to **ON**.

Enable the **Terminate the call after playing emergency messages listed below** check box if you want the **Emergency** block to end the call after playing the emergency prompts.

If you want to store the result of the emergency flag in a variable, select a variable from the list.



Scenarios

If you want to:

- Enable emergency mode:
  - Open your application.
  - Locate the **Emergency** block.
  - Toggle the emergency mode switch to **ON**.

- Control emergency mode from a web service:
  - Add an HTTP REST block in the **Initialize** phase.
  - Assign relevant output to a variable (for example, `varEmergency = true`).
  - In the **Self Service** phase, add a Segmentation block.
  - Add a condition/branch (`varEmergency == true`).
  - Add an **Emergency** block in this segment.
  - Set the emergency mode switch to **ON** permanently in this block.
  - Specify any emergency prompts.
  - Enable or disable the **Terminate the call after playing emergency messages listed below** check box.
  - Specify a variable to store the result of the emergency flag check.

Special Day Block

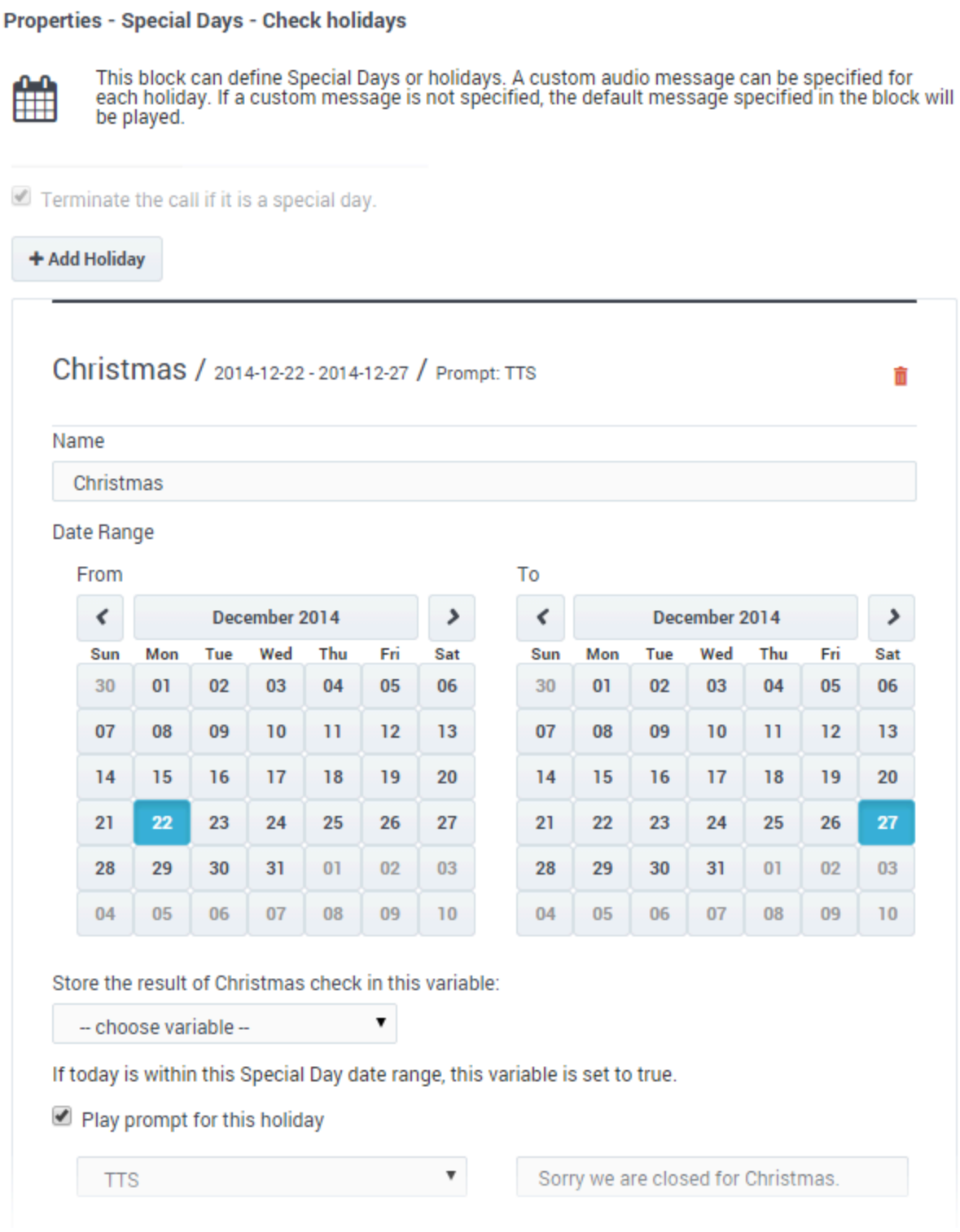
You can use the **Special Day** block in the **Initialize**, **Self Service**, or **Assisted Service** phase to define holidays and other special days, and play prompts to announce closures or greetings. It can also terminate the call if your business is closed.

Holiday tab

Click **Add Holiday** to add a holiday. A holiday entitled **New** appears in the list.

Next, click the **New** holiday to edit its settings. Configure the following options:

- In the **Date Range** section, use the provided calendars to select the **From** and **To** dates for the holiday.
- Assign a variable to the **Store the result of *Special Day Name* in this variable** property if you want to use the result of this check later in application. If specified, the variable is assigned the Boolean value of true or false.
- Enable the **Play prompt for this holiday** check box to play a special greeting to callers during a holiday.


Properties - Special Days - Check holidays
This block can define Special Days or holidays. A custom audio message can be specified for each holiday. If a custom message is not specified, the default message specified in the block will be played.
Terminate the call if it is a special day.
+ Add Holiday
Christmas / 2014-12-22 - 2014-12-27 / Prompt: TTS
Name
Christmas
Date Range
From
To
December 2014
Sun Mon Tue Wed Thu Fri Sat
30 01 02 03 04 05 06
07 08 09 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 01 02 03
04 05 06 07 08 09 10
Store the result of Christmas check in this variable:
-- choose variable --
If today is within this Special Day date range, this variable is set to true.
Play prompt for this holiday
TTS
Sorry we are closed for Christmas.

Default Prompts tab

Click **Add Prompt** to specify a prompt to play if the application receives a call on a special day, and that particular day does not have a custom prompt.

Results tab

You can select a variable that will be set to **true** if any of the special days listed in the block evaluate to **true**.

You can also select a variable that will be set to **true** if the special days evaluation processing completed correctly. If it did not, it will be set to **false**.

Scenarios

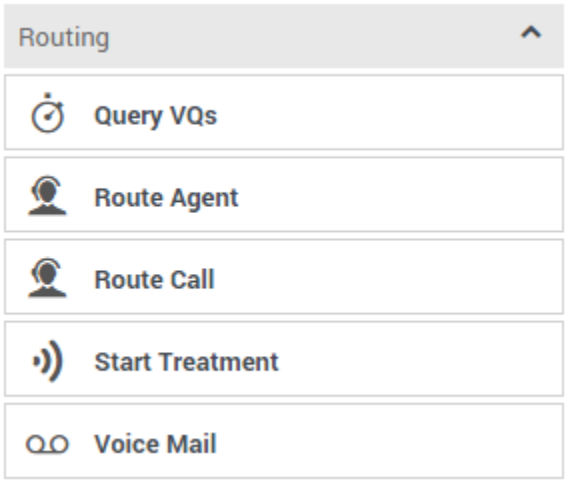
If you want to:

- Play a special greeting during Thanksgiving.
  - Click **Add Holiday** and set the **From** and **To** dates.
  - Enable the **Play prompt for this holiday** check box.
  - Select **TTS** and enter text to speak, or select **Announcement** to choose a predefined announcement.
- Play the same greeting for all holidays.
  - Click **Add Holiday** and create one or more holidays.
  - Do not enable the **Play prompt for this holiday** check box.
  - In the **Default Prompts** tab, add prompts to the table.

Routing Blocks

These blocks specify where the call should be *routed* when certain conditions are met.

You might not see all of the blocks listed here on your Palette. The blocks shown depend on the features that are enabled and the type of application that is being built. For example, the **Route Digital** block is only available for Digital application types.



Use the links below to learn more about each block.

Query VQs

Queries virtual queues and assigns their Estimated Wait Times.

Used in: **Initialization**, **Assisted Service**

Route Agent

Specifies routing to a particular agent.

Used in: **Assisted Service**

Route Call

Specifies routing to an agent based on various criteria.

Used in: **Assisted Service**

Start Treatment

Plays uninterrupted audio to callers while their call is being routed.

Used in: **Assisted Service**

Voice Mail

Routes calls to voicemail.

Used in: **Assisted Service**

Transfer

Transfers a call to another destination.

Used in: **Self Service**

Route Digital

(Digital application types only) Routes a multimedia interaction to a target.

Used in: **Assisted Service**

Query VQs Block

You can use the **Query VQs** block in the **Initialize** or **Assisted Service** phases to determine the Estimated Wait Time for several virtual queues.

Before a call is queued, the **Query VQs** block can check the Estimated Wait Time for all targeted virtual queues to help determine which virtual queue receives the call. The results are stored in user variables.

> **Important**
>
> If the query isn't able to obtain a result from a targeted virtual queue, the value of the Estimated Wait Time is set to zero (0).

Using this Block

In the block properties, click **Add Assignment**.

Next, select the **Virtual Queue** to query and the **Variable** to which Designer assigns the Estimated Wait Time value for that virtual queue.

Choose whether to save the Estimated Wait Time in **minutes** or **seconds**.

In the drop-down menus, select the variable to store the **name** of the virtual queue with the lowest Estimated Wait Time value and the variable to store the **value** of the lowest Estimated Wait Time.

Properties - Query VQs

This block can query several virtual queues and assign the value of the estimated wait time for the virtual queue to a variable.

Specify virtual queues to query for estimated wait time.

+ Add Assignment

Virtual Queue	Variable	Delete
Bronze	ewt1	
Silver	ewt2	
FD_Billing_Gold	ewt3	

Estimated Wait Time will be saved in: Minutes

Store the name of the virtual queue with the lowest estimated wait time in this variable.

vq_best_ewt

Store the value of the lowest estimated wait time in this variable.

ewt_best

Route Agent Block

You can use the **Route Agent** block in the **Assisted Service** phase to route calls to an agent based solely on:

- a specified **Agent ID** and **Virtual Queue**
- a specified **Agent Login** and **SIP Switch**, or
- the **Last Called Agent** that the caller spoke to

To route calls based on skills and other criteria, use the Route Call block.

You can sequentially place multiple **Route Agent** blocks with different settings, so that if routing fails in one block, your application proceeds to the next block. When a **Route Agent** block successfully routes the call to an agent, the application moves to the **Finalize** phase, ignoring any subsequent blocks in the **Assisted Service** phase.

Agent Routing tab

Agent Routing

Select a Routing type:

- **Agent ID** - Select the **Agent ID** (specified as a variable) and **Virtual Queue** that the call will be routed to.
- **Agent Login** - Select the **Agent Login** (specified as a variable) and **SIP Switch** (where the selected agent is logged on, also specified as a variable) that the call will be routed to.
- **Last Called Agent** - Select this option to route the call to the agent that the caller last spoke with.

Other Routing Settings

- **Overall timeout** - If enabled, you can specify how long the application must wait before moving on to the next block. Optionally, you can enable the check box to specify a variable.
- **Route only if the Agent is ready** - If enabled, a call is routed to an agent only if his status is set to Ready. If disabled, the call is routed to an agent regardless of his status.

Treatments tab

Specify a busy treatment to execute while waiting for an agent to become available. You can choose to play audio and/or execute a shared module.

- Learn more about busy treatments

> **Important**
>
> After a busy treatment has been executed at least 10 times, Designer exits the **Route Agent** block and moves to the next block if the average duration of the treatment is less than 1000 ms (for example, due to a missing audio file).

Audio

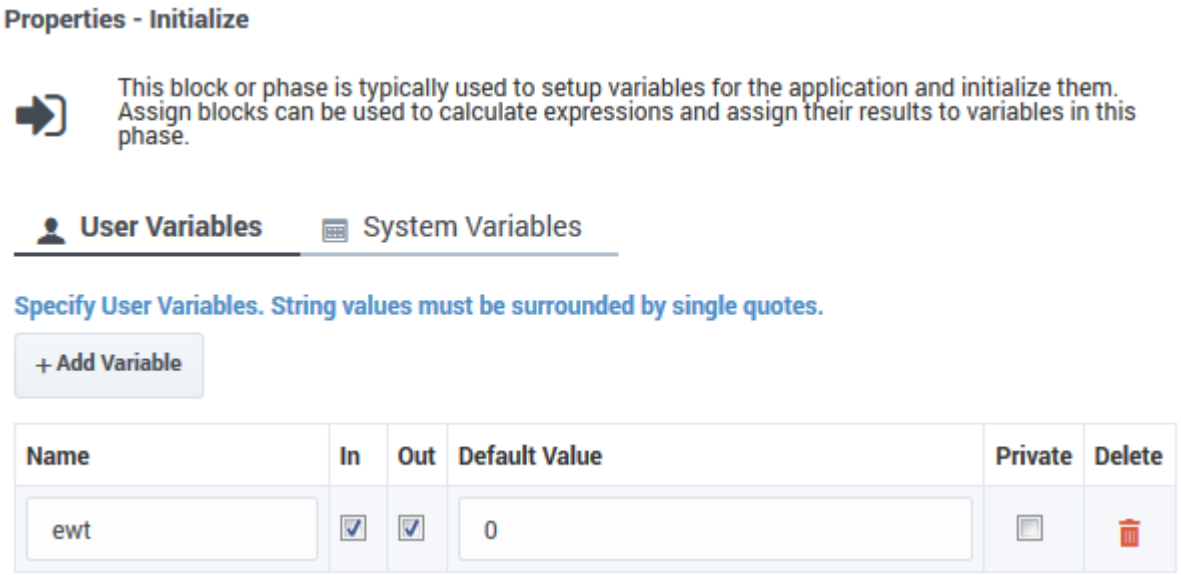
Click **Add Audio** to add a Play Message child block underneath this **Route Agent** block. The collection of audio plays repeatedly until the call is successfully routed or times out.

Shared Module

Click **Add Module** to add a Shared Module child block underneath this **Route Agent** block. In the child block, you can select a shared module to execute.

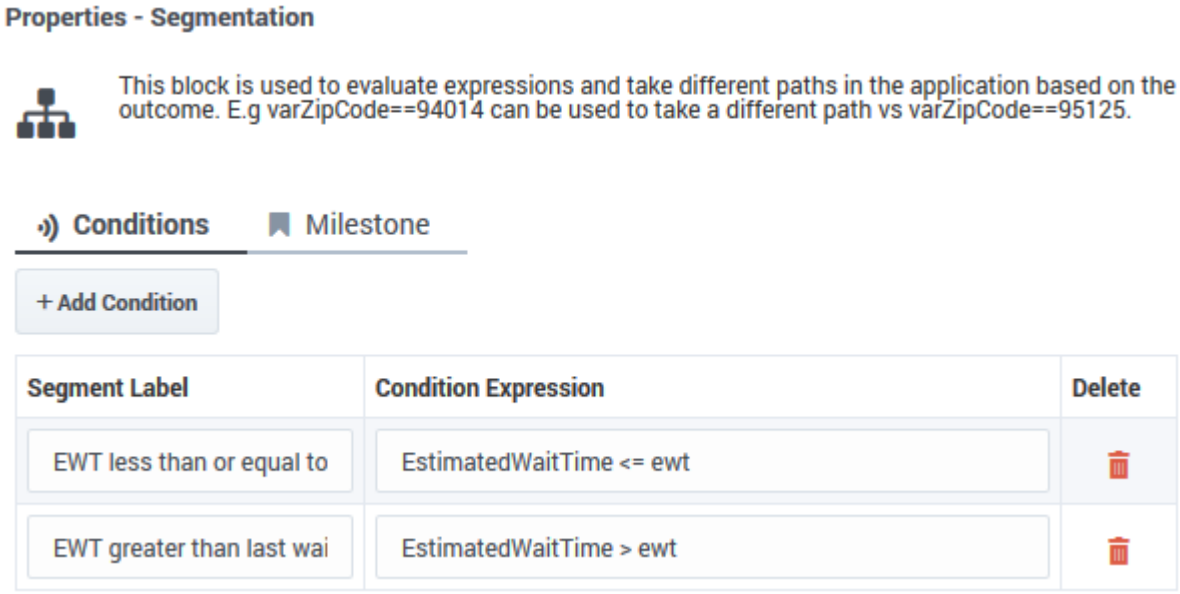
A potential use case is to execute a shared module based on a specified set of conditions that can change over time and respond to external factors. For example, you might use a shared module that can play one announcement for callers if the estimated wait time (EWT) is beyond a certain threshold, and another announcement for when they are the next caller in the queue. To set up this feature:

1. In the application, create a user variable, `ewt`, and set its default value to `0`.
2. Create a **Self Service** type shared module.
3. In the shared module, create a user variable, `ewt`, and set its default value to `0`.

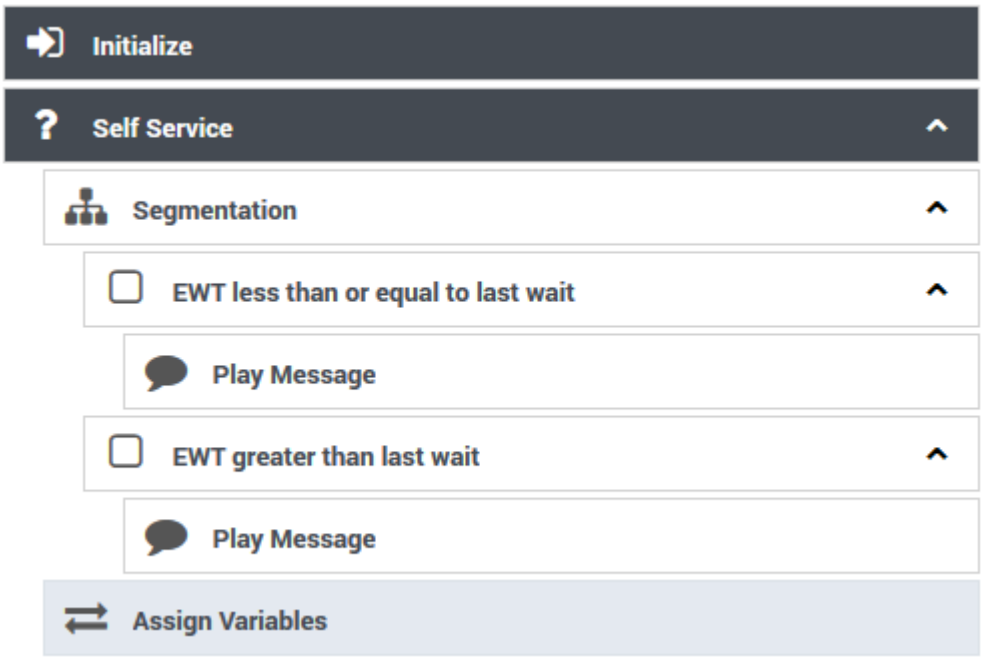


4. In the **Self Service** phase of the shared module, add a **Segmentation** block. Add the conditions as

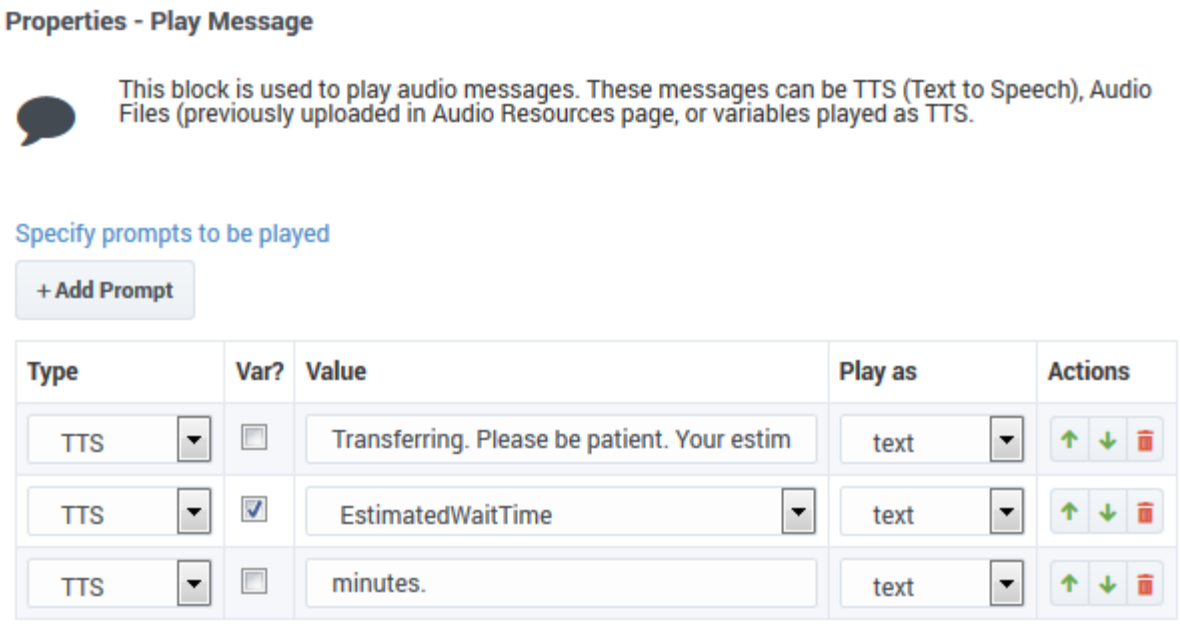
shown below:



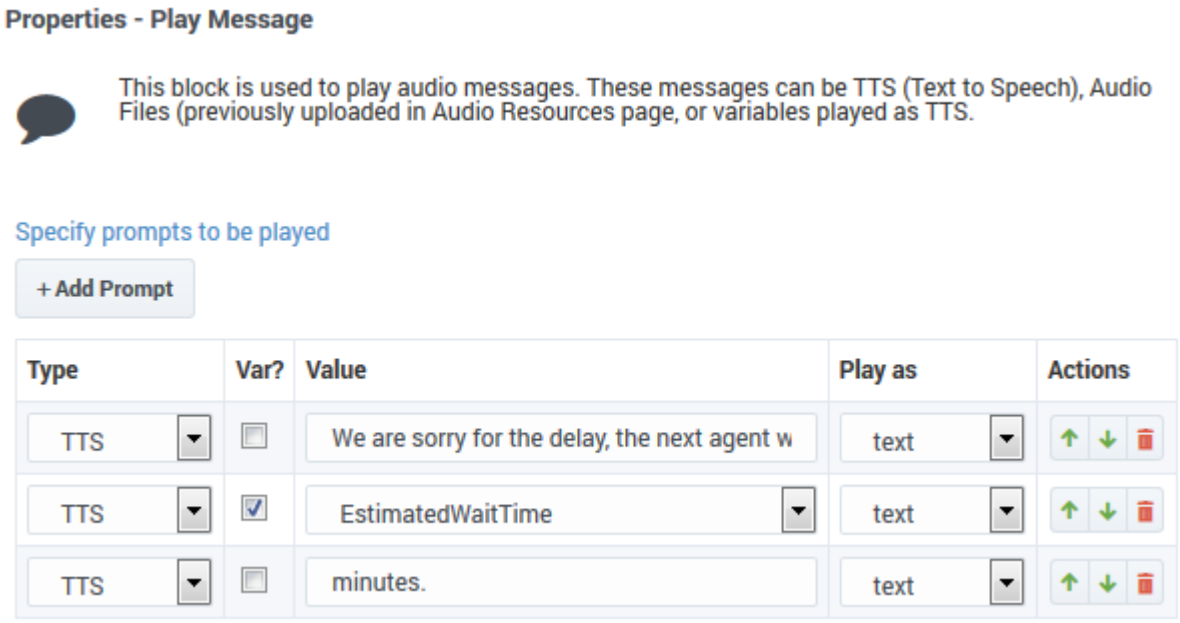
5. Add two **Play Message** blocks as child blocks of the condition blocks, and add an **Assign Variables** block at the end. Your shared module should appear as shown below:



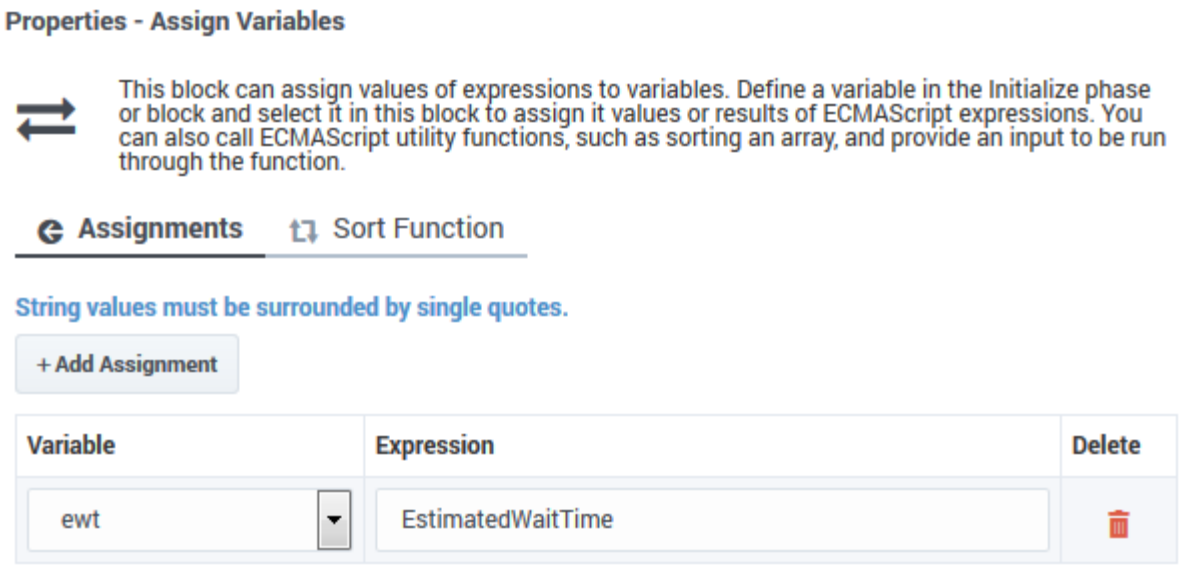
6. Configure the first **Play Message** block. An example is below:



7. Configure the second **Play Message** block. An example is below:



8. Configure the **Assign Variables** block as shown below:



9. In your application, select the **Route Agent** block and click the **Treatments** tab.
10. Click **Add Module**. A child **Shared Module** block appears beneath the **Route Agent** block.
11. In the child **Shared Module** block, select the shared module that you created in Step 2.

The application passes **ewt** to the shared module, along with the system variables, which includes **EWT**. The shared module compares **ewt** and **EWT** in the **Segmentation** block and executes a **Play Message** block, depending on which variable is larger. At the end, the shared module sets **ewt** to **EWT** before returning to the application.

Advanced tab

Targeting

Use Agent Status

If checked, the routing engine uses the **status** flag to route the call to an agent. If not checked, the routing engine uses the **loggedin** flag to route the call to an agent.

> **Important**
> This check box has no effect if you enabled the **Route only if the Agent is ready** check box in the **Agent Routing** tab.

Threshold Expression

This option enables you to use an ECMAScript (or JavaScript) expression to further refine a routing threshold for the specified target(s). Threshold expressions for the **Route Agent** block can be used for the following routing types:

- Agent ID
- Agent Login
- Last Called Agent

Threshold expressions can contain variables or reference queue-specific values, such as *sdata(target, statistic)* or *callage()*. Strings must be enclosed in single quotes. For example:

For more information about using ECMAScripts in Designer, see Assigning values to variables.

> **Important**
> For routing types that have multiple targets, the script defined in **Threshold Expression** applies to all targets.

Extensions

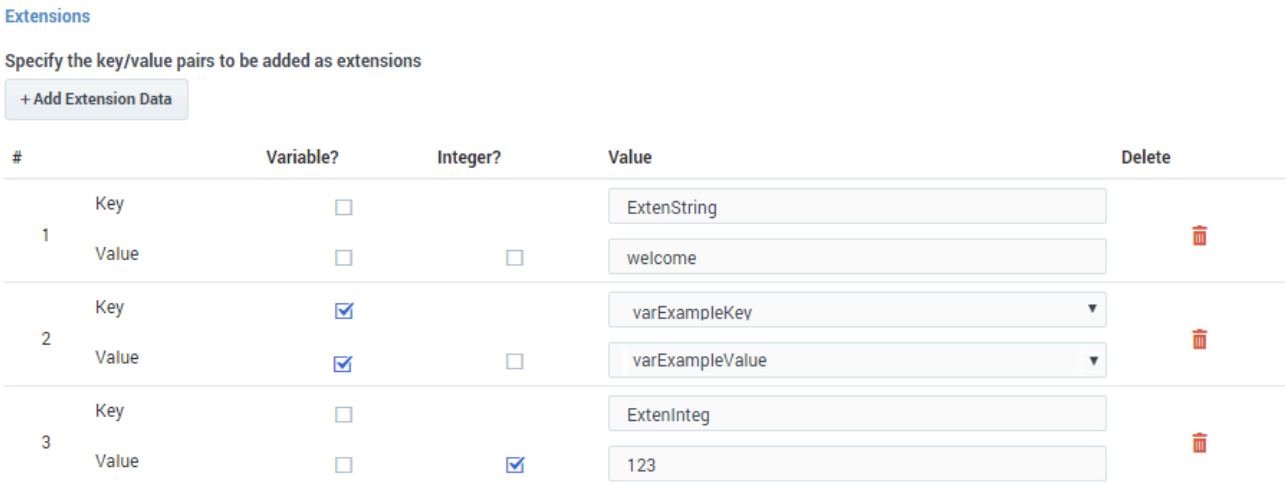
Click **Add Extension Data** to add an extension as a key-value pair to this block. The value type can be a string or integer.

If you want to use a variable for the **Key** or **Value**, select the **Variable** checkbox and then select a variable from the drop-down menu. If the **Value** is an integer, select the **Integer** checkbox.

> **Important**
> You do not need to enclose extension values in quotes. However, if the quote is part of the value, you must escape the quote character by using a preceding backslash. For example:
>
> - Incorrect: `Joe's Pizza`
> - Correct: `Joe\'s Pizza`

> **Important**
>
> Designer displays an error message if Extension Data is added, but the **Key** and **Value** settings are not defined.

This example shows a few different ways that key-value pairs can be added as extensions:



Results tab

Select a variable to store the result of this **Route Agent** block execution.

Route Call Block

You can use the **Route Call** block in the **Assisted Service** phase to route calls to an agent based on various criteria, such as Skills and Agent Groups.

You can sequentially place multiple **Route Call** blocks with different settings, so that if routing fails in one block, your application proceeds to the next block. When a **Route Call** block successfully routes the call to an agent, the application moves to the **Finalize** phase, ignoring any subsequent blocks in the **Assisted Service** phase.

Call Routing tab

Select Routing Type section

Choose between the following routing options:

Skill based routing with relaxing criteria

Routes the call to an agent that has the required skills. If selected, you can choose from the following options:

- **Use system variables RoutingTarget / RoutingVirtualQueue set already in Menu Options** - Use system variables that were set in a Menu Option block.
- **Specify Skills in this block** - Specify one or more skills and a Virtual Queue to use to route this call. If you specified more than one skill, you can choose whether the routing engine considers any or all of the selected skills:
  - **all skills** - The application must use all of the selected skills to route the call.
  - **any skill** - The application can use any of the selected skills to route the call.

> **Important**
>
> This option uses the skill level specified in the **Use Skill Proficiency level** setting (documented below). For example, if you set an initial skill level of 8, Designer only routes the call to agents with the specified skills that have a level of 8 or greater. You cannot set an individual level for each specified skill.

- **Use Skill Proficiency level** - Enter a Skill level. The call is routed to an agent that has a skill level equal to or higher to the value you provide. If you enable **Reduce skill requirements**, the required skill level is gradually decremented by a specified skill level, until it reaches the specified minimum skill level. This option allows you to expand the group of agents that can receive this call if other agents are busy.

Skill expression based routing

Enter a skill expression in the **Skill Expression** tab, or click the drop-down menu to select a variable that specifies a skill expression.

Agent Group routing

Route the call to a specific Agent Group or a variable that holds the name of an Agent Group at runtime.

Agent routing

Route the call to agents by using a variable that holds the ID of an agent at runtime. You must use the following format: *agentid*`@`*optional_statserver*`.A`.
Example: `1001@StatServer.A`.

Campaign Group routing

Route the call to a specific Campaign Group or a variable that holds the name of a Campaign Group at runtime.

Route Point routing

Transfer a call to another Routing Point. The application associated to that Routing Point is responsible for handling the call. You can select a known Routing Point or a variable that holds the name of a Routing Point at runtime.

Direct number routing

Transfer a call to a number. You can use a variable to hold the number to use at runtime or add direct number elements. Specify the weight for each number and Designer displays and uses the percentage ranking based on the weightings.

Force Route

Force the call to route to a direct number. When using this routing type, Designer routes the call in a way that is similar to how an interaction is redirected by Route Point routing. When selected, you can specify the target as a *literal* value, or as a variable that holds a *string*, *number*, or *object* value.

> **Important**
>
> - The **Routing Priority** tab and the **Targeting Options** in the **Advanced** tab (**Clear targets from queue if this block times out** and **Early exit from this block if no agents are logged in**) are not applicable when using **Force Route**.
> - When the **Force Route** option is selected, the overall timeout for the **Route Call** block is limited to 30 seconds.

Other Routing Settings section

Routing Algorithm

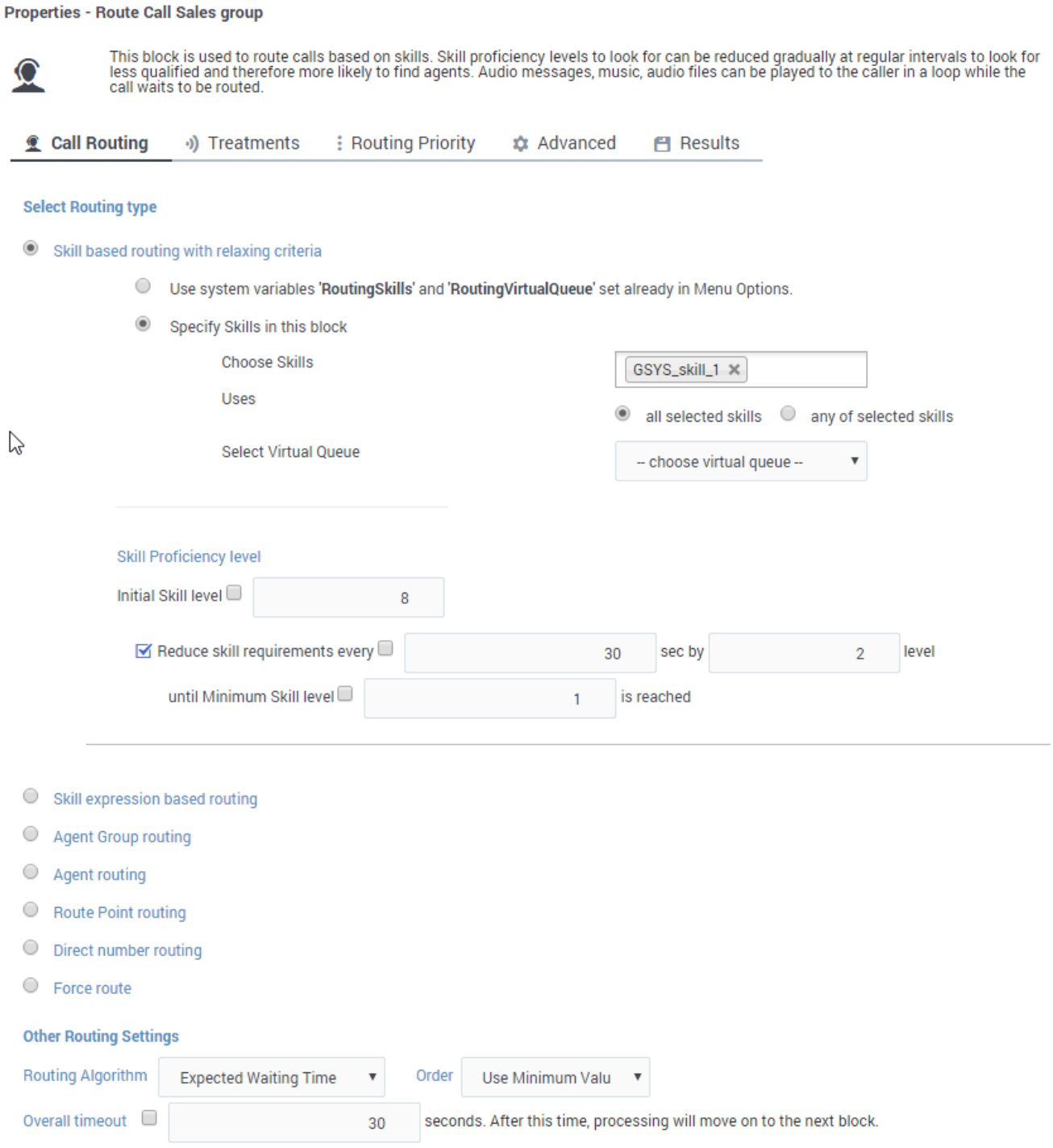
Select which algorithm is used to choose an agent when more than one agent is available. For more information, see Routing Algorithms.

Overall timeout

Enter the maximum time (in seconds) to wait for an agent to be available before moving to the next block. Optionally, you can enable the check box to specify a variable.

> **Important**
>
> System variables **SelectedTarget**, **SelectedVirtualQueue**, **SelectedComponent**, **SelectedTargetObject**, **SelectedAgent**, and **Access** are automatically set when the call is routed to an agent and can be used later in the application. Refer to the **Initialize** phase's **System Variables** tab to read a detailed description for each of these variables.

Example


Properties - Route Call Sales group
This block is used to route calls based on skills. Skill proficiency levels to look for can be reduced gradually at regular intervals to look for less qualified and therefore more likely to find agents. Audio messages, music, audio files can be played to the caller in a loop while the call waits to be routed.
Call Routing
Treatments
Routing Priority
Advanced
Results
Select Routing type
Skill based routing with relaxing criteria
Use system variables 'RoutingSkills' and 'RoutingVirtualQueue' set already in Menu Options.
Specify Skills in this block
Choose Skills
GSYS_skill_1
Uses
all selected skills
any of selected skills
Select Virtual Queue
-- choose virtual queue --
Skill Proficiency level
Initial Skill level
8
Reduce skill requirements every
30
sec by
2
level
until Minimum Skill level
1
is reached
Skill expression based routing
Agent Group routing
Agent routing
Route Point routing
Direct number routing
Force route
Other Routing Settings
Routing Algorithm
Expected Waiting Time
Order
Use Minimum Valu
Overall timeout
30
seconds. After this time, processing will move on to the next block.

Skill Expression tab

> **Important**
>
> This tab only appears if you selected the **Skill expression based routing** option in the **Call Routing** tab.

If you selected the option **Skill expression based routing** in the **Call Routing** tab, you must build the skill expression to identify the best agent to handle the call. The skill expression consists of a list of skills for which you must individually set an operator and an integer value.

Arrange individual skill conditions in the conditions sets. You can specify skills by name or variables that contain the name of the skills at runtime.

> **Important**
>
> When using **Skill expression based routing** and you are building the entire skill expression within a variable, you must manually add the single quotes around the skill names.
>
> For example, use this:
>
> ```
> "'New iPhone' > 7"
> ```
>
> instead of this:
>
> ```
> "New iPhone > 7"
> ```

Treatments tab

Specify a busy treatment to execute while waiting for an agent to become available. You can choose to play audio and/or execute a shared module.

- Learn more about busy treatments

> **Important**
>
> After a busy treatment has been executed at least 10 times, Designer exits the **Route Call** block and moves to the next block if the average duration of the treatment is less than 1000 ms (for example, due to a missing audio file). (However, this does not apply if the **Force Route** option is selected.)

Audio

Click **Add Audio** to add a Play Message child block underneath this **Route Call** block. The collection of audio plays repeatedly until the call is successfully routed or times out.

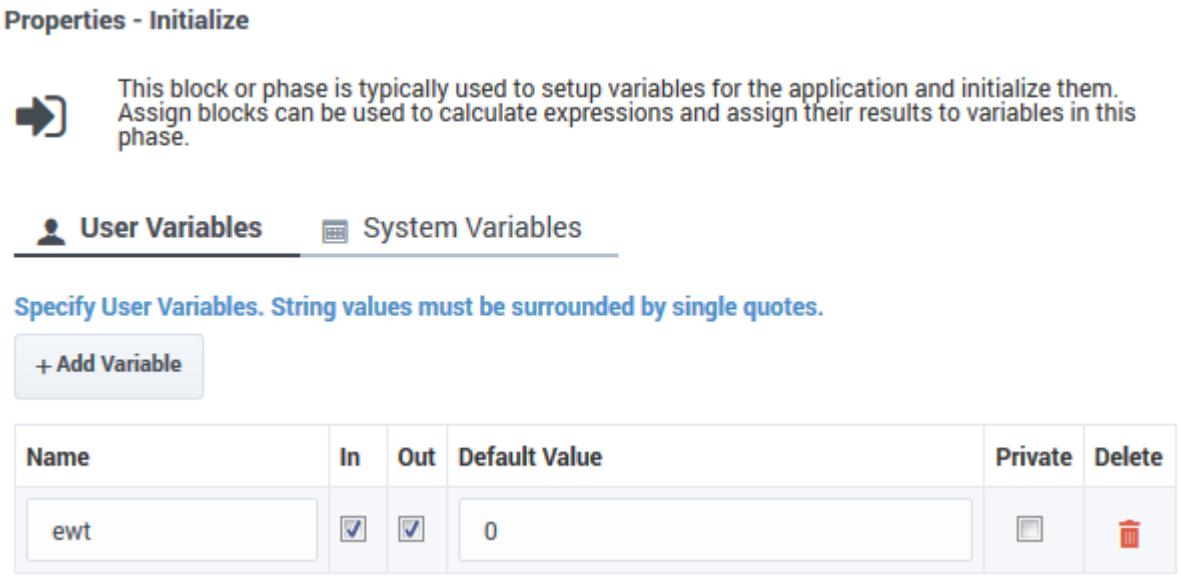
Shared Module

Click **Add Module** to add a Shared Module child block underneath this **Route Call** block. In the child block, you can select a shared module to execute.

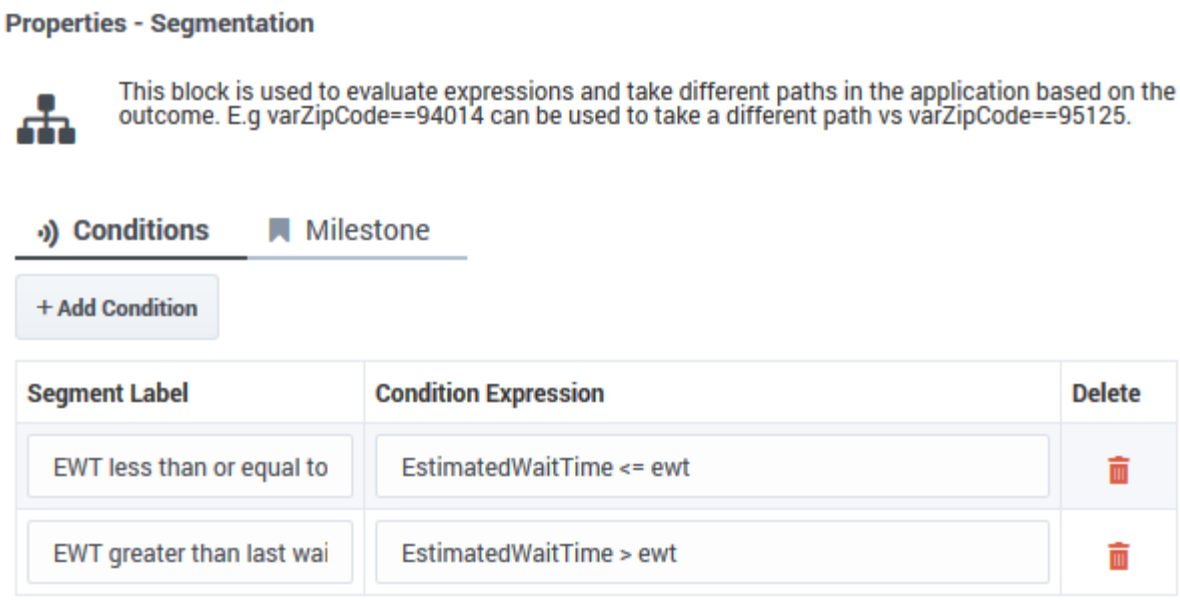
A potential use case is to execute a shared module based on a specified set of conditions that can change over time and respond to external factors. For example, you might use a shared module that can play one announcement for callers if the estimated wait time (EWT) is beyond a certain threshold, and another announcement for when they are the next caller in the queue.

To set up this feature:

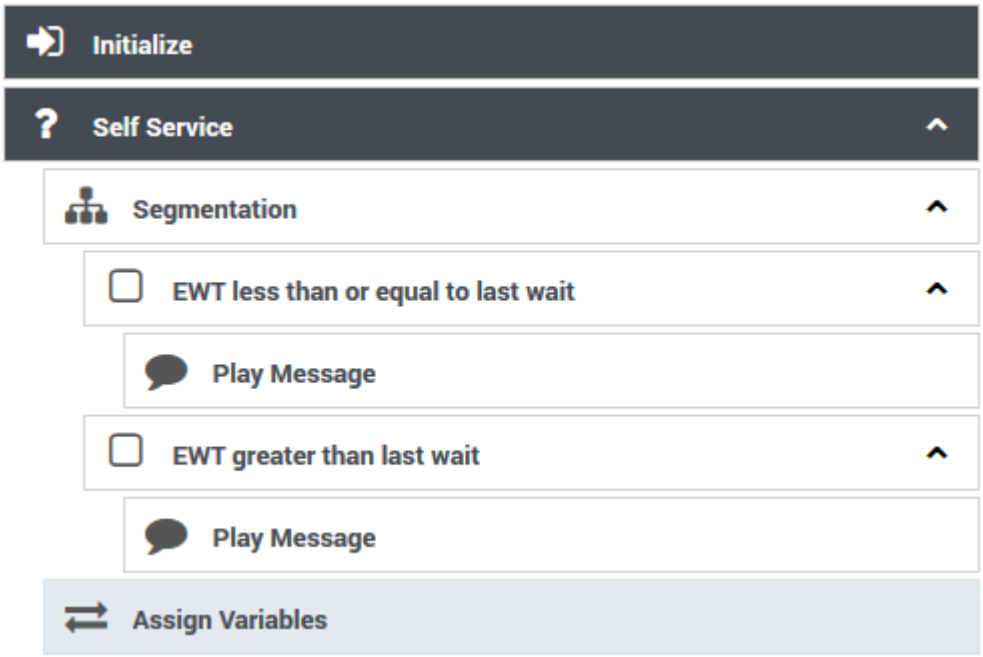
1. In the application, create a user variable, `ewt`, and set its default value to `0`.
2. Create a **Self Service** type shared module.
3. In the shared module, create a user variable, `ewt`, and set its default value to `0`.



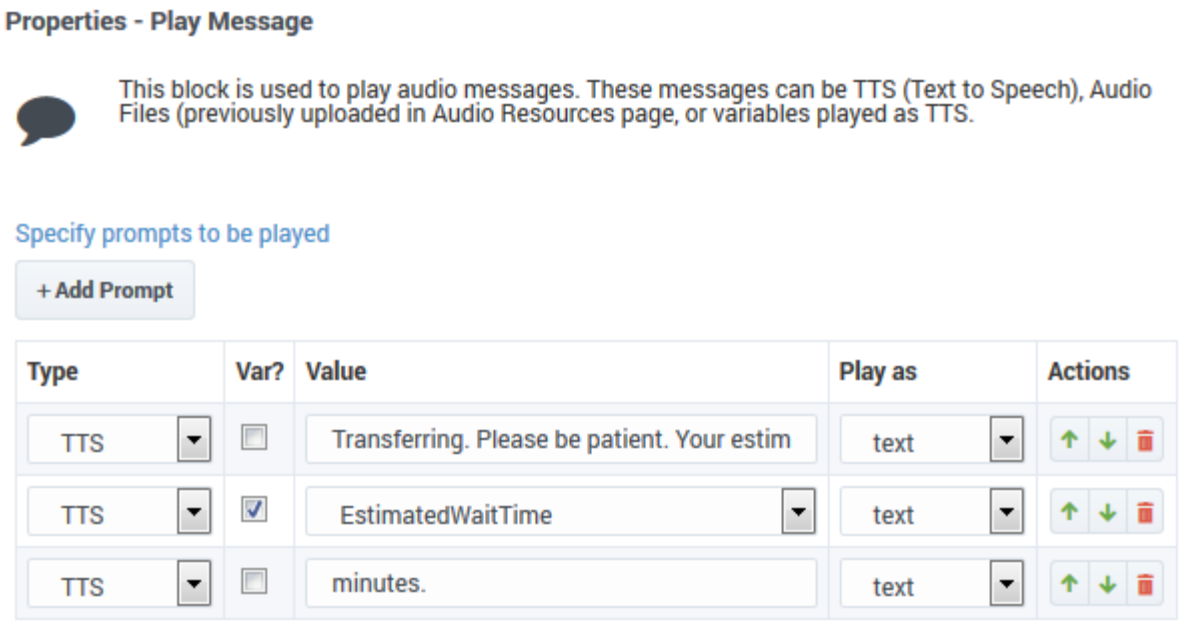
4. In the **Self Service** phase of the shared module, add a **Segmentation** block. Add the conditions as shown below:



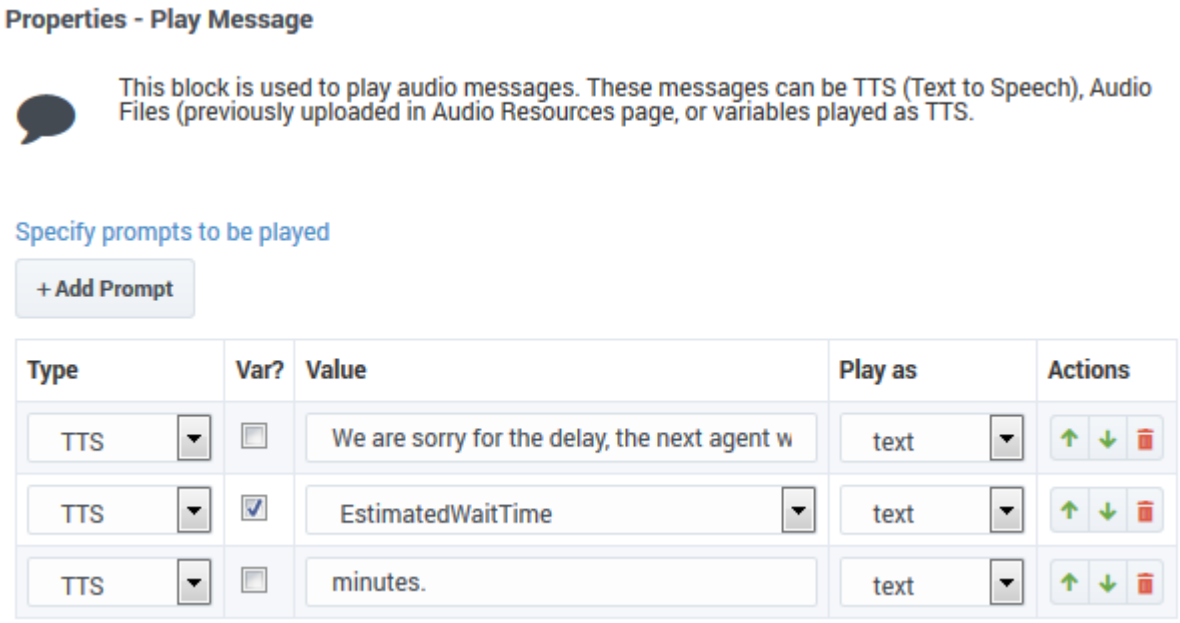
5. Add two **Play Message** blocks as child blocks of the condition blocks, and add an **Assign Variables** block at the end. Your shared module should appear as shown below:



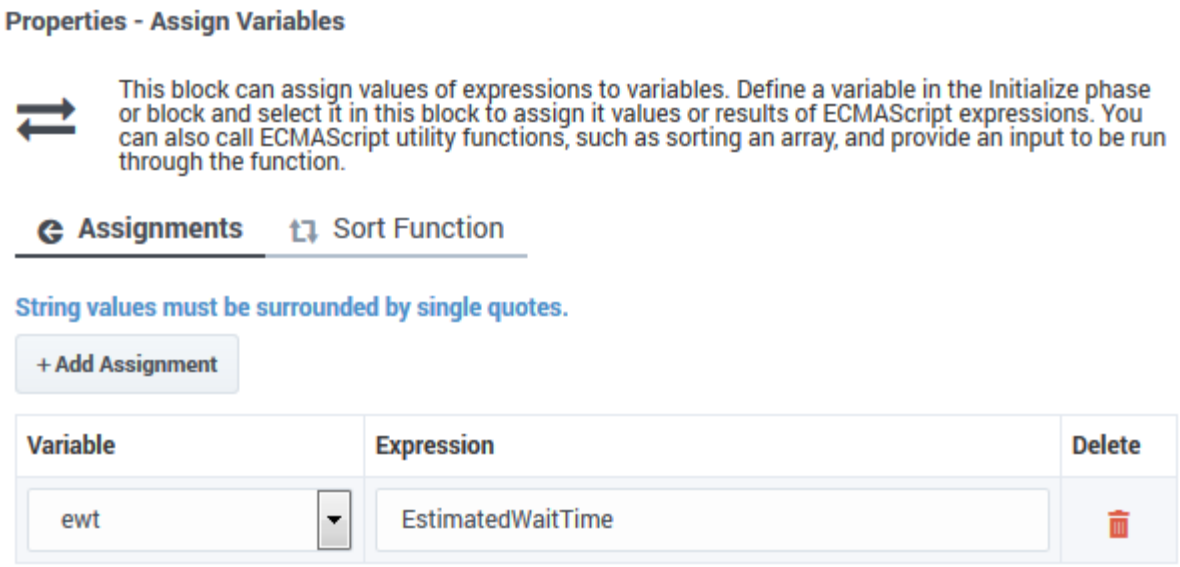
6. Configure the first **Play Message** block. An example is below:



7. Configure the second **Play Message** block. An example is below:



8. Configure the **Assign Variables** block as shown below:



9. In your application, select the **Route Call** block and click the **Treatments** tab.
10. Click **Add Module**. A child **Shared Module** block appears beneath the **Route Call** block.
11. In the child **Shared Module** block, select the shared module that you created in Step 2.

The application passes **ewt** to the shared module, along with the system variables, which includes **EWT**. The shared module compares **ewt** and **EWT** in the **Segmentation** block and executes a **Play Message** block, depending on which variable is larger. At the end, the shared module sets **ewt** to **EWT** before returning to the application.

Routing Priority tab

Use Priority during Routing

Enable this check box to use priority-based routing, which prioritizes your calls depending on your business requirements.

To prioritize calls, you must segment calls and assign the name of that segment to a variable. You must select this variable in the **Lookup Priority table based on this variable** drop-down menu.

You can customize this table with your own segment definitions to fit your business needs. If the specific segment is not found, then the value specified for **Initial priority** is used. Enter a value in **Increment size** to increase the priority of a call that remains in a queue over time. The priority increment is defined for each segment, but a default increment is configurable with the **Increment Size** property.

Increment Priority every ___ seconds

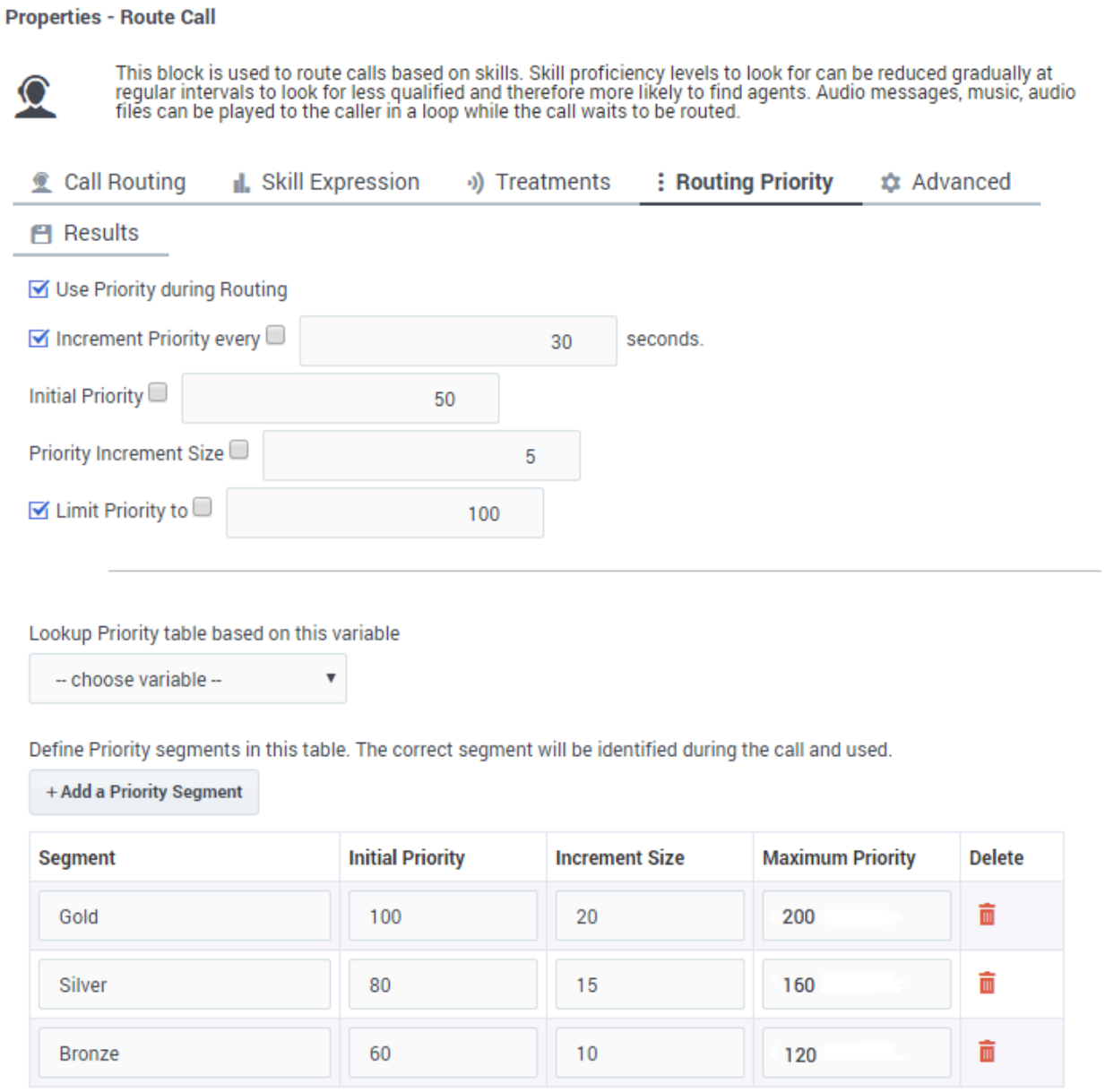
Enable this check box to specify the time interval between priority increments. If you enable the other check box beside the field, you can select a variable that specifies the overall **Routing**

Timeout and **Priority Increment Interval** properties.

Limit Priority to

If the **Increment Priority every ___ seconds** option is enabled, you can use this option to set a maximum priority value. For example, if the initial priority is `50`, you can use this option to not let the priority value increase beyond `100`, as shown here:

Example



If you enable the other check box beside the field, you can select a variable for this option.

Advanced tab

Targeting section

Clear targets from queue if this block times out

Enable this check box to specify whether the pending request for a target should be kept active or not after exiting this block on timeout. When the request is kept active (check box is disabled), an agent may be selected after the block times out if, for example, an agent with the matching criteria is ready after the block was exited.

Early exit from this block if no agents are logged in

Enable this check box to exit the block if no agents are logged in for the selected routing target (such as Agent or Agent Group, skill expression based, or skill based routing with relaxing criteria).

Route only to local agents

If you have selected **Skill based routing with relaxing criteria** or **Skill expression based routing**, you can enable this option. When enabled, the call is routed to a local agent who matches the target skill.

> **Tip**
>
> If you want to route to local agents as the preferred option, but then route to all agents if there are no local agents available with the required skill, you can set up cascaded routing.
>
> Here's a way you can do that:
>
> - Set up the **Route Call** block with **Route only to local agents** enabled, a short **Overall timeout** property value, and **Clear targets from queue if this block times out** deselected.
> - Then, set up any **Route Call** blocks that are further down the application flow with **Route only to local agents** not selected.
>
> You can watch this video to see a short demonstration of how to set this up.
>
> Link to video
>
> You might also want to modify skill relaxing settings to run faster on routing blocks that target local agents.

Threshold Expression

This option enables you to use an ECMAScript (or JavaScript) expression to further refine a routing threshold for the specified target(s). Threshold expressions for the **Route Call** block can be used for the following routing types:

- Skill
- Skill Expression
- Agent Group
- Agent
- Campaign Group
- Direct Number

Threshold expressions can contain variables or reference queue-specific values, such as *sdata(target, statistic)* or *callage()*. Strings must be enclosed in single quotes. For example:

For more information about using ECMAScripts in Designer, see Assigning values to variables.

> **Important**
>
> For routing types that have multiple targets (such as Agent Group or Agent), the script defined in **Threshold Expression** applies to all targets.

Greetings section

Enable the check box beside **Customer Greeting** and/or **Agent Greeting** to play an audio file to that person while the call is being connected.

For customers, you might use this feature to play a legal disclaimer, or to announce that the call might be recorded (if you use call recording in your contact center). For agents, you might use a variable to announce the customer name or other relevant information.

After you enable **Customer Greeting** and/or **Agent Greeting**, you can select an audio file to play by clicking the icon in the **Announcement** field. This is useful for customer greetings that play a static disclaimer audio file.

Optionally, enable the **Var?** check box to use a variable to dynamically select the audio file. This is useful for agent greetings that use a variable to provide call-specific information, such as the customer name.

Note that:

- The **Customer Greeting** plays continuously until the **Agent Greeting** finishes playing.
- When the **Customer Greeting** and **Agent Greeting** contain different prompt values, each prompt is played to the customer and the agent as specified.
- When only one option contains a value, the same prompt is played to both the customer and the agent.
- If the **Customer Greeting** or **Agent Greeting** cannot be played, the customer is immediately connected to the agent. No greetings are played.

Extensions section

Click **Add Extension Data** to add an extension as a key-value pair to this block. The value type can be a string or integer.

If you want to use a variable for the **Key** or **Value**, select the **Variable** checkbox and then select a variable from the drop-down menu. If the **Value** is an integer, select the **Integer** checkbox.

Important

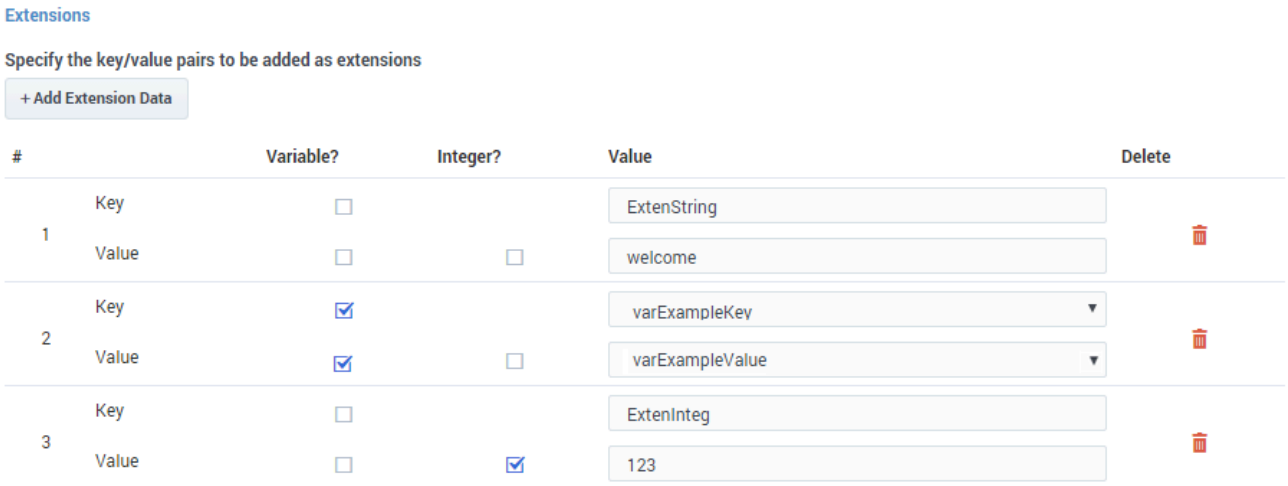
You do not need to enclose extension values in quotes. However, if the quote is part of the value, you must escape the quote character by using a preceding backslash. For example:

- Incorrect: `Joe's Pizza`
- Correct: `Joe\'s Pizza`

Important

Designer displays an error message if Extension Data is added, but the **Key** and **Value** settings are not defined.

This example shows a few different ways that key-value pairs can be added as extensions:



Results tab

Select a variable in the **Store selected agent ID in this variable** drop-down menu to keep track in a specific variable the ID of the agent selected as a result of this **Route Call** block execution. The **SelectedAgent** system variable is transparently assigned this same agent ID value.

You can also select a variable in the **Store the outcome of the Route Call block in this variable** drop-down menu to store the result of this **Route Call** block execution.

Routing Algorithms

The Route Call and Route blocks allow you to select certain statistic types that can be used as routing algorithms. (For more information about statistic types, see Statistic Types on the Statistic block page.)

Agent Loading

Selects agents within an agent group and calculates a vector based on three values:

- The number of busy DNs.
- The agents' time in Ready state (the same value as **Time In Ready State**)
- A random number.

Agent Occupancy

Enables the routing engine to route interactions to the least occupied agent (the agent with the lowest occupancy rate). The occupancy rate is determined by the ratio between the time the agent has been busy since last login compared to the agent’s total login time.

Agent Occupancy enables the routing engine to evaluate multiple available agents and select the least occupied agent, which balances the workload among available agents.

This statistic is defined and calculated when the agent logs in and can be used only in statistics that are applied for an agent. This statistic cannot be used with agent groups.

How it works

Consider the following scenario:

- After login, Agent 1 was on a call for five minutes and was in a Ready state for five minutes. His occupancy is 50 per cent (5 / (5+5)).
- After login, Agent 2 was on a call for one minute and was in a Ready state for two minutes. His occupancy is 33 per cent (1 / (1+2)).

When using **Agent Occupancy**, the routing engine distributes an incoming interaction to Agent 2, as he is the least occupied agent.

Expected Waiting Time

Provides wait-in-queue estimates for the last interaction that entered a virtual queue. This statistic has been designed for the multimedia model, but assumes agents cannot handle more than one simultaneous non-voice interaction at a time.

This statistic applies to all types of media and is calculated and refreshed internally for the last 600 seconds.

Load Balance

Load balancing between routing targets helps to ensure there is an equal distribution of interactions among queues, DNs, agent groups and queue groups. This statistic considers the number of calls distributed between objects, the percentage of busy agents, the expected waiting time, or other routing features.

When this statistic is defined, the routing engine automatically counts calls that are routed to different DNs and queues and adjusts the logic so there is an equal distribution of interactions across the specific DNs or queues.

Most Skilled Agent

This routing algorithm finds the most skilled agent among all available agents who match the skills selected in the Route Call (or Route) block.

How it works

Consider the following scenario:

- A **Route Call** block has three skills selected.
- Agent A has skill levels of 7/7/7 for those skills.
- Agent B has skill levels of 10/5/5, respectively.

The algorithm finds the agent with the highest *combined* skill among the selected skills. Therefore, if both agents are available, the algorithm selects Agent A as the most skilled agent.

> **Important**
>
> This routing algorithm only works with *skill*-based routing; *skill expression*-based routing is not supported.

Randomly

Controls the mechanism for selecting a default target. If the target list has more than one available target, this option functions as follows:

- If this option is set to **random**, the routing engine picks a target according to its internal rules.
- If this option is set to **FIFO**, the routing engine picks the first available target in the list.

Round Robin

Quickly selects a target from an agent group by applying a round-robin algorithm. The routing engine searches through a list of agents in the target object and returns the first available agent (Agent 1). For the next call, the routing engine bypasses the previously selected agent and picks the next available agent.

When the routing engine reaches the end of the list of agents, it returns to the beginning of the list.

Time in Ready State

Provides the total current time of an agent in Ready state. It is calculated based on availability - that is, there is no activity currently in progress on a particular DN and the agent has placed the DN in Ready state. **Time in Ready State** aggregates the total time for the state and this information can be used to build the target list.

You can use **Time in Ready State** to route interactions to the agent that has been waiting in Ready state for the longest time.

Route Digital Block

If your application is a Digital type, you can use the **Route Digital** block in the **Assisted Service** phase to route interactions to an agent based on various criteria, such as skills.

You can sequentially place multiple **Route Digital** blocks with different settings, so that if routing fails in one block, your application proceeds to the next block.

When a **Route Digital** block successfully routes the interaction to an agent, the application moves to the **Finalize** phase, ignoring any subsequent blocks in the **Assisted Service** phase.

Click here to watch a video showing an example of skill-based routing for chat interactions.

Routing Tab

Select Routing type

Choose between the following routing options:

Skill based routing with relaxing criteria

Routes the interaction to an agent that has the required skills. If selected, you can choose from the following options:

- **Use system variables *RoutingSkills* and *RoutingVirtualQueue* set already in Menu Options** - Use system variables that were set in a Menu Option block.
- **Specify Skills in this block** - Specify one or more skills and a Virtual Queue to use to route this interaction. If you specified more than one skill, you can choose whether the routing engine considers **any** or **all** of the selected skills.

> **Important**
>
> This option uses the skill level specified in the **Skill Proficiency level** setting (documented below). For example, if you set an initial skill level of 8, Designer only routes the interaction to agents with the specified skills that have a level of 8 or greater. You cannot set an individual level for each specified skill.

- **Skill Proficiency level** - Enter the initial and minimum skill levels. The interaction is routed to an agent that has a skill level equal to or higher to the values provided. If you enable **Reduce skill requirements every**, you can have the skill level decremented by a certain amount every *x* number of seconds, until the minimum skill level is reached. This option allows you to expand the group of agents that can receive this interaction if other agents are busy.

Skill expression based routing

Enter a skill expression in the **Skill Expression** tab, or click the drop-down menu to select a variable that specifies a skill expression.

Application/Queue routing

Route the interaction to a specific application or queue.

Agent Group routing

Route the interaction to an Agent Group.

Agent routing

Route the interaction to agents by using a variable that holds the ID of an agent at runtime. You must use the following format: *agentid*`@`*optional_statserver*`.A`. Example: `1001@StatServer.A`.

Other Routing Settings

Routing Algorithm

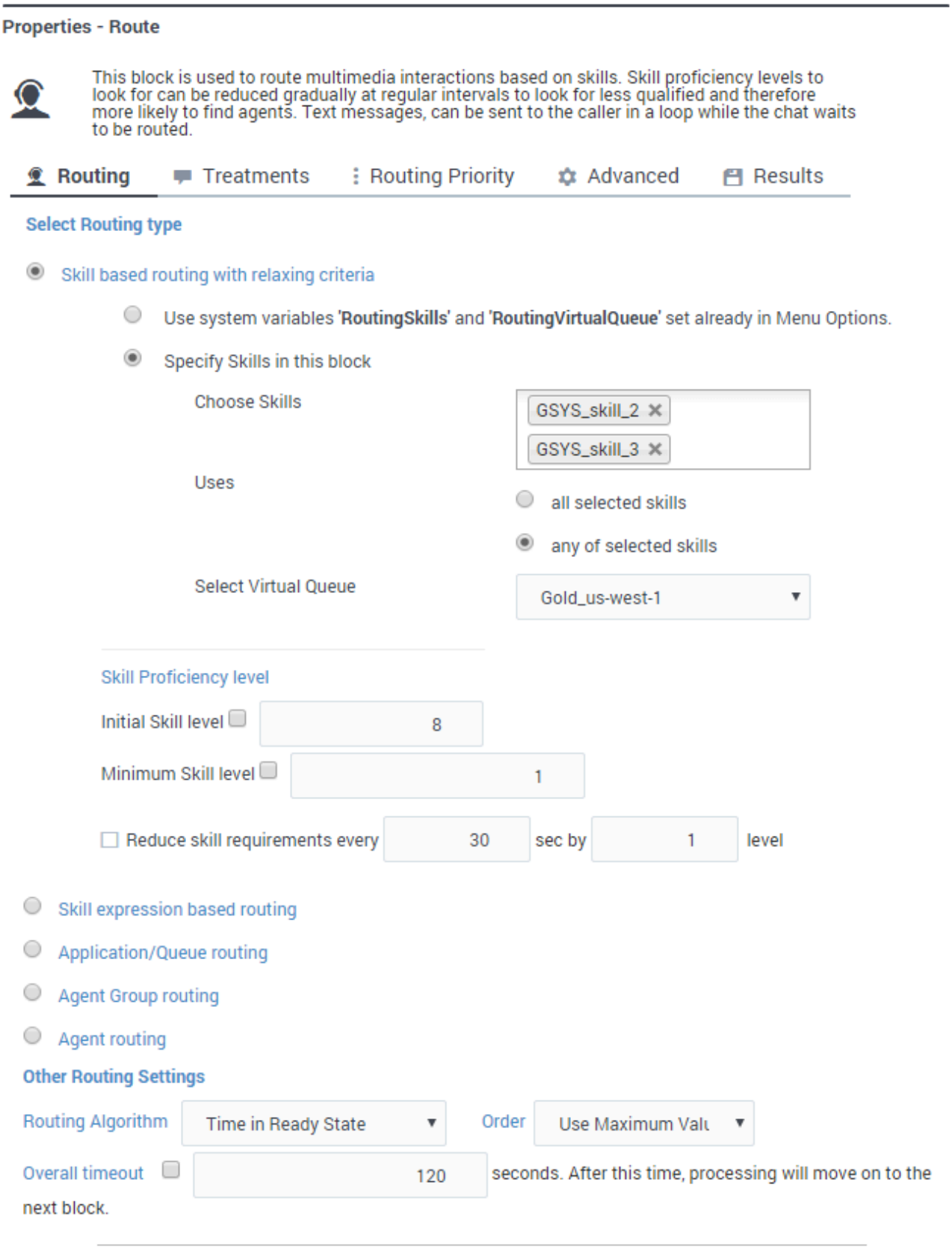
Select which algorithm is used to choose an agent when more than one agent is available. (For more information about the routing algorithms, see Statistic Types on the **Statistic** block page.)

Overall timeout

Enter the maximum time (in seconds) to wait for an agent to be available before moving to the next block. Optionally, you can enable the check box to specify a variable.

> **Important**
>
> System variables **SelectedTarget**, **SelectedVirtualQueue**, **SelectedComponent**, **SelectedTargetObject**, **SelectedAgent**, and **Access** are automatically set when the interaction is routed to an agent and can be used later in the application. Refer to the **Initialize** phase's **System Variables** tab to read a detailed description for each of these variables.

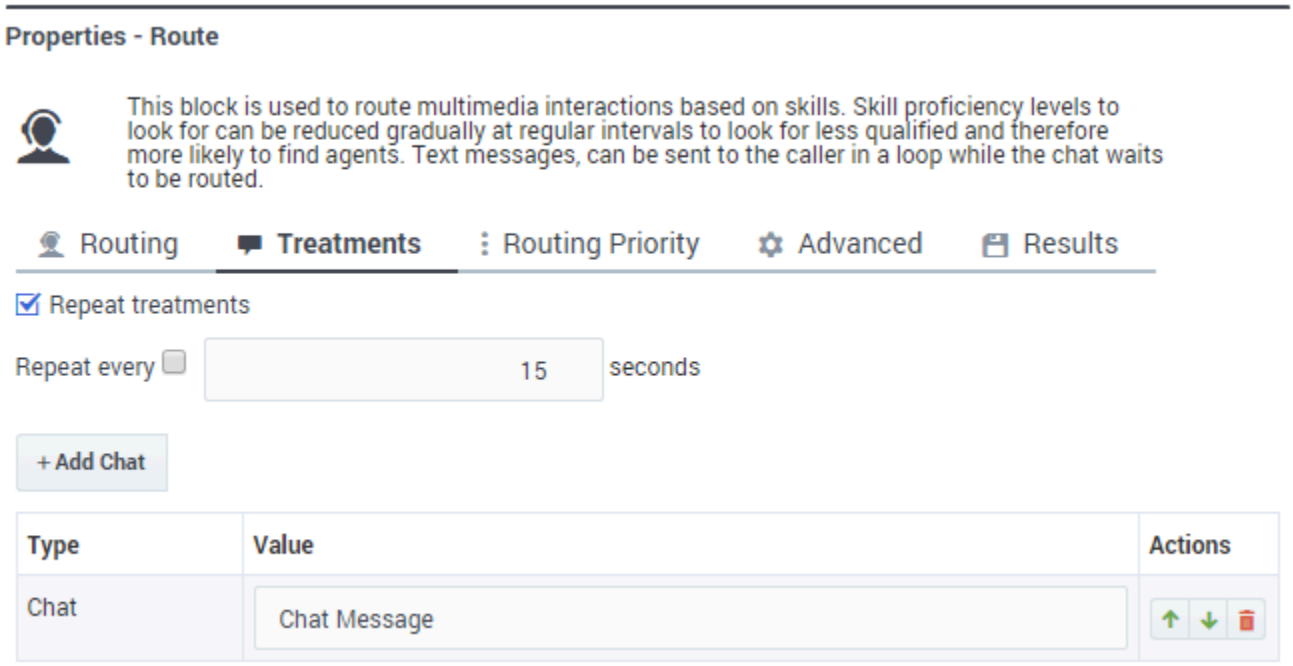
Example


Properties - Route
This block is used to route multimedia interactions based on skills. Skill proficiency levels to look for can be reduced gradually at regular intervals to look for less qualified and therefore more likely to find agents. Text messages, can be sent to the caller in a loop while the chat waits to be routed.
Routing
Treatments
Routing Priority
Advanced
Results
Select Routing type
Skill based routing with relaxing criteria
Use system variables 'RoutingSkills' and 'RoutingVirtualQueue' set already in Menu Options.
Specify Skills in this block
Choose Skills
GSYS_skill_2
GSYS_skill_3
Uses
all selected skills
any of selected skills
Select Virtual Queue
Gold_us-west-1
Skill Proficiency level
Initial Skill level
8
Minimum Skill level
1
Reduce skill requirements every
30
sec by
1
level
Skill expression based routing
Application/Queue routing
Agent Group routing
Agent routing
Other Routing Settings
Routing Algorithm
Time in Ready State
Order
Use Maximum Valu
Overall timeout
120
seconds. After this time, processing will move on to the next block.

Treatments tab

Use this tab to specify a busy treatment to execute while waiting for an agent to become available. The busy treatment can be a **Chat** message that you can set to repeat at regular intervals.

Example



Routing Priority tab

Use Priority during Routing

Enable this check box to use priority-based routing, which prioritizes your interactions depending on your business requirements.

To prioritize interactions, you must segment interactions and assign the name of that segment to a variable. You must select this variable in the **Lookup Priority table based on this variable** drop-down menu.

You can customize this table with your own segment definitions to fit your business needs. If the specific segment is not found, then the value specified for **Initial priority** is used. Enter a value in **Increment size** to increase the priority of an interaction that remains in a queue over time. The priority increment is defined for each segment, but a default increment is configurable with the **Increment Size** property.

Increment Priority every ___ seconds

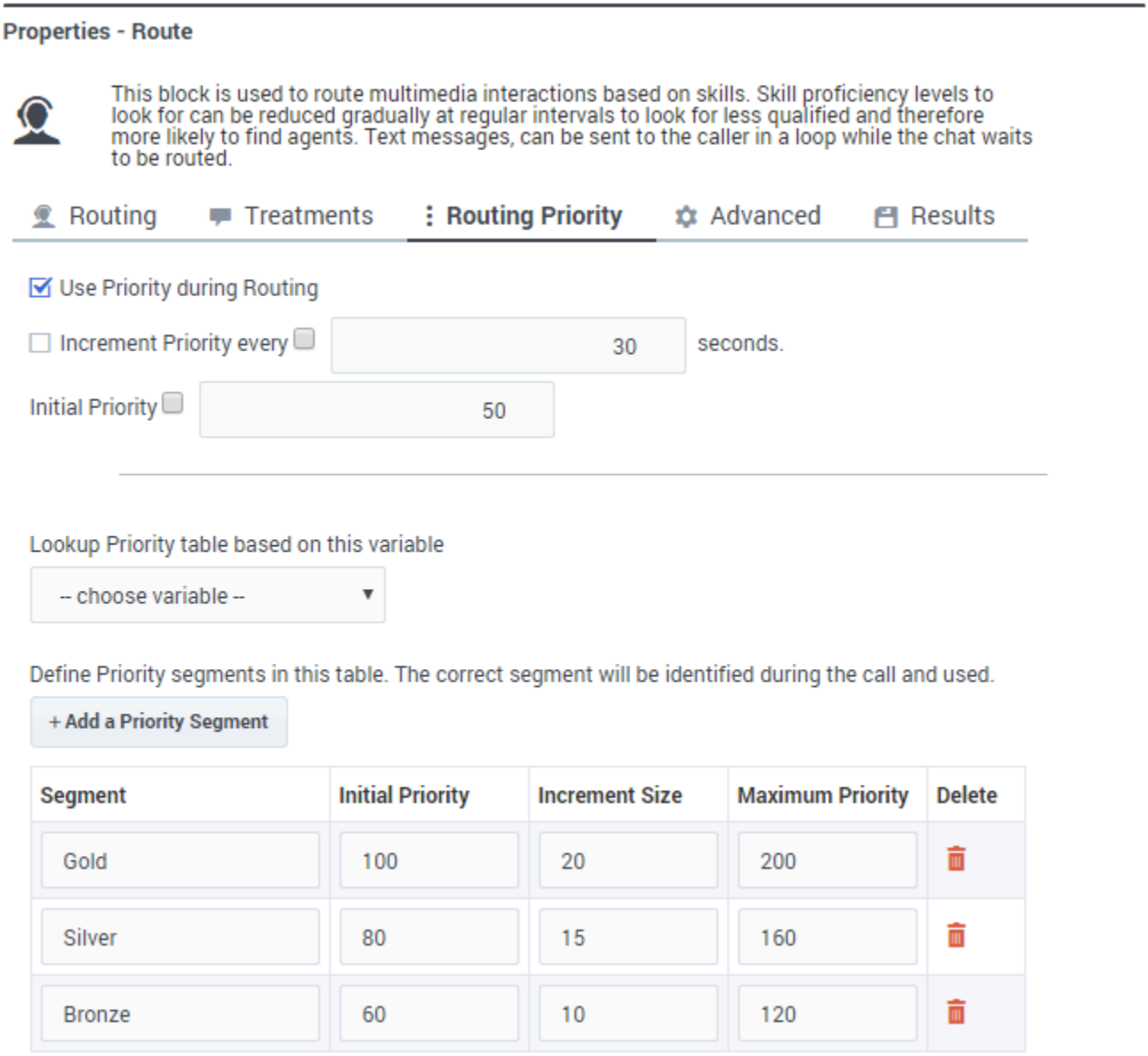
Enable this check box to specify the time interval between priority increments. If you enable the other check box beside the field, you can select a variable that specifies the overall **Routing Timeout** and **Priority Increment Interval** properties.

Limit Priority to

If the **Increment Priority every ___ seconds** option is enabled, you can use this option to set a maximum priority value. For example, if the initial priority is `50`, you can use this option to not let the priority value increase beyond `100`.

If you enable the other check box beside the field, you can select a variable for this option.

Example


Properties - Route
This block is used to route multimedia interactions based on skills. Skill proficiency levels to look for can be reduced gradually at regular intervals to look for less qualified and therefore more likely to find agents. Text messages, can be sent to the caller in a loop while the chat waits to be routed.
Routing
Treatments
Routing Priority
Advanced
Results
Use Priority during Routing
Increment Priority every
30
seconds.
Initial Priority
50
Lookup Priority table based on this variable
-- choose variable --
Define Priority segments in this table. The correct segment will be identified during the call and used.
+ Add a Priority Segment
Segment
Initial Priority
Increment Size
Maximum Priority
Delete
Gold
100
20
200
Silver
80
15
160
Bronze
60
10
120

Advanced tab

Targeting

Clear targets from queue if this block times out

Enable this check box to specify whether the pending request for a target should be kept active or not after exiting this block on timeout. When the request is kept active (check box is disabled), an agent may be selected after the block times out if, for example, an agent with the matching criteria is ready after the block was exited.

Early exit from this block if no agents are logged in

Enable this check box to exit the block if no agents are logged in for the selected routing target (such as Agent or Agent Group, skill expression based, or skill based routing with relaxing criteria).

Threshold Expression

This option enables you to use an ECMAScript (or JavaScript) expression to further refine a routing threshold for the specified target(s). Threshold expressions for the **Route** block can be used for the following routing types:

- Skill
- Skill Expression
- Agent Group
- Agent
- Last Called Agent

Threshold expressions can contain variables or reference queue-specific values, such as *sdata(target, statistic)* or *callage()*. Strings must be enclosed in single quotes. For example:

For more information about using ECMAScripts in Designer, see Assigning values to variables.

> **Important**
> For routing types that have multiple targets (such as Agent Group or Agent), the script defined in **Threshold Expression** applies to all targets.

Route only to local agents

If you have selected **Skill based routing with relaxing criteria** or **Skill expression based routing**, you can enable this option. When enabled, the call is routed to a local agent who matches the target skill.

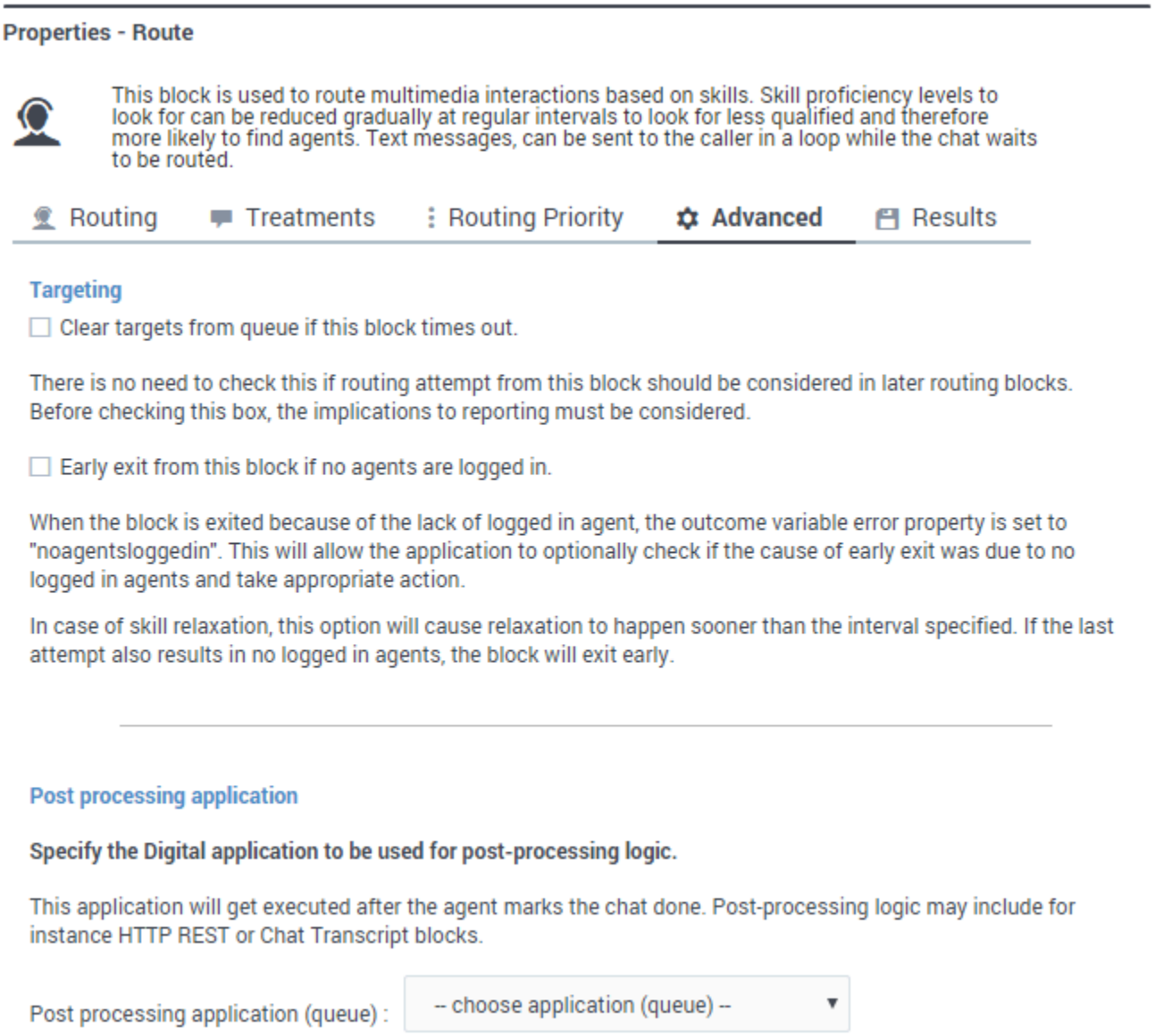
> **Tip**
>
> If you want to route to local agents as the preferred option, but then route to all agents if there are no local agents available with the required skill, you can set up cascaded routing.
>
> Here's a way you could do that:
>
> - Set up the **Route** block with **Route only to local agents** enabled, a short **Overall timeout** property value, and **Clear targets from queue if this block times out** deselected.
> - Then, set up any **Route** blocks that are further down the application flow with **Route only to local agents** not selected.
>
> You can watch this video to see a short demonstration of how to set this up. (The video demonstrates this using a Route Call block, but the steps are the same.)
>
> Link to video
>
> You might also want to modify skill relaxing settings to run faster on routing blocks that target local agents.

In the **Post processing application** section, you can specify the digital application (queue) that will be used for post-processing logic. This application will be executed after the agent marks the chat as *done*.

Post-processing logic may include, for example, HTTP REST or Chat Transcript blocks.

> **Tip**
>
> If the purpose of your post-processing application is to only send a Chat Transcript, Genesys recommends that you use a Terminate block in the post-processing application to prevent it from sending of multiple copies of the transcript.

Example



Results tab

Select a variable in the **Store selected agent ID in this variable** drop-down menu to keep track in a specific variable the ID of the agent selected as a result of this **Route** block execution. The **SelectedAgent** system variable is transparently assigned this same agent ID value.

You can also select a variable in the **Store the outcome of the Route block in this variable** drop-

down menu to store the result of this **Route** block execution.

Videos

This video shows an example of using skill-based routing to route chat interactions.

Link to video

Start Treatment Block

You can use the **Start Treatment** block to play uninterrupted audio (a "busy treatment") to callers while their calls are being processed by more than one Route Call or Route Agent blocks.

Typically, busy treatments are played by the Route Call or Route Agent block, but the audio stops playing when the flow moves on to the next block.

- Learn more about busy treatments

> **Important**
>
> After a busy treatment has been executed at least 10 times, Designer exits the Route Call or Route Agent block and moves to the next block if the average duration of the treatment is less than 1000 ms (for example, due to a missing audio file).

With the **Start Treatment** block, callers won't hear any breaks in the audio as their call is being routed. The audio will continue to play until another treatment is started (for example, the flow reaches another **Start Treatment** block, or **Play Message** is started by one of the routing blocks), the call is routed, or the Assisted Service phase ends.

> **Tip**
>
> Remember that when you start a new treatment, it immediately stops any treatment that is running.

Module tab

Use this tab to select the Shared Module that will play the audio file.

Is Synced

- If enabled, the **Start Treatment** block will remain active while the treatment is running. The application will not move to the next block in the phase until a condition specified in the Navigation tab causes the application to jump to another block.

> **Important**
>
> When exiting the block on a navigation condition, the treatment keeps playing (like with an

asynchronous block) although the **Start Treatment** block is exited.

- If not enabled, the **Start Treatment** block is exited as soon as the treatment starts, and processing moves to the next block in the phase.

> **Important**
>
> When **Is Synced** is not enabled, an asynchronized treatment can continue to play after the **Start Treatment** block has been exited, and will continue playing until a new treatment is started or the Assisted Service phase ends.

Signature tab

This tab displays any **Input Parameters** and **Output Parameters** that are returned by the Self Service Shared Module running the busy treatment.

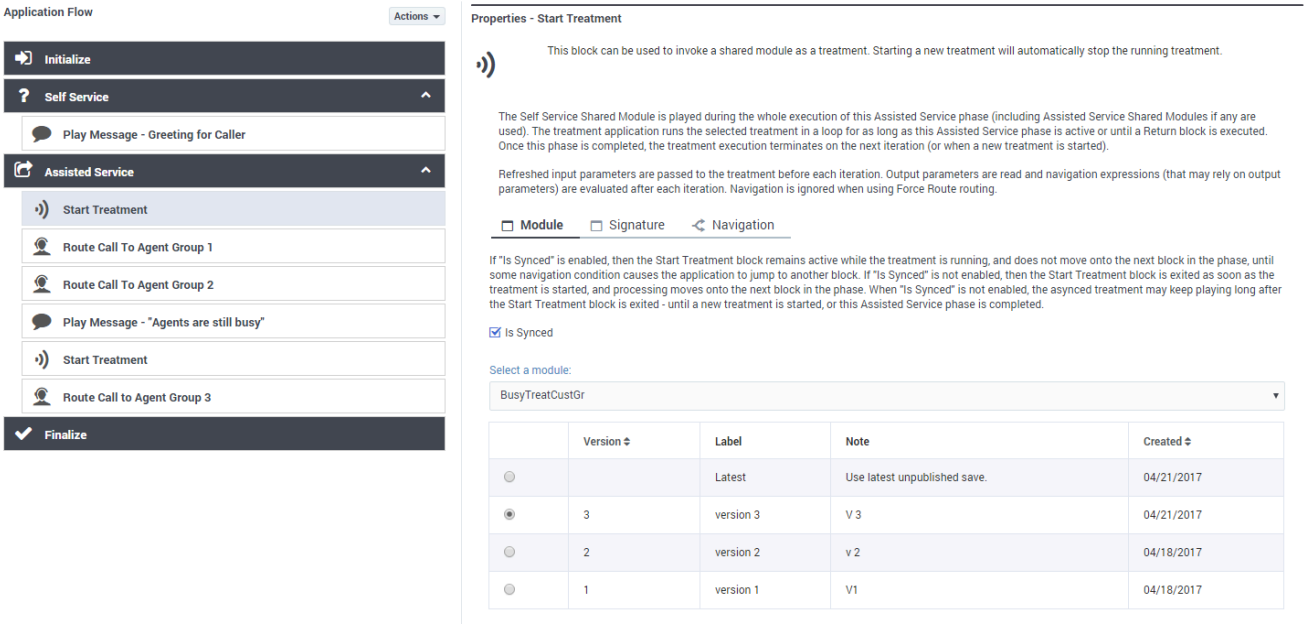
Navigation tab

Use this tab to add a **Condition Expression** that will redirect the application to the selected target block. You can select a target by **Name**, **Type**, or **Description**.

> **Important**
>
> The busy treatment will continue to play to callers during the redirect, as the target block is part of the same Assisted Service phase as the **Start Treatment** block.

Example

Here's a look at how this block can be used in an application flow (click to enlarge):


Application Flow
Actions
Initialize
Self Service
Play Message - Greeting for Caller
Assisted Service
Start Treatment
Route Call To Agent Group 1
Route Call To Agent Group 2
Play Message - "Agents are still busy"
Start Treatment
Route Call to Agent Group 3
Finalize
Properties - Start Treatment
This block can be used to invoke a shared module as a treatment. Starting a new treatment will automatically stop the running treatment.
The Self Service Shared Module is played during the whole execution of this Assisted Service phase (including Assisted Service Shared Modules if any are used). The treatment application runs the selected treatment in a loop for as long as this Assisted Service phase is active or until a Return block is executed. Once this phase is completed, the treatment execution terminates on the next iteration (or when a new treatment is started).
Refreshed input parameters are passed to the treatment before each iteration. Output parameters are read and navigation expressions (that may rely on output parameters) are evaluated after each iteration. Navigation is ignored when using Force Route routing.
Module
Signature
Navigation
If "Is Synced" is enabled, then the Start Treatment block remains active while the treatment is running, and does not move onto the next block in the phase, until some navigation condition causes the application to jump to another block. If "Is Synced" is not enabled, then the Start Treatment block is exited as soon as the treatment is started, and processing moves onto the next block in the phase. When "Is Synced" is not enabled, the asynced treatment may keep playing long after the Start Treatment block is exited - until a new treatment is started, or this Assisted Service phase is completed.
Is Synced
Select a module:
BusyTreatCustGr
Version
Label
Note
Created
Latest
Use latest unpublished save.
04/21/2017
3
version 3
V 3
04/21/2017
2
version 2
v 2
04/18/2017
1
version 1
V1
04/18/2017

Transfer Block

> **Important**
>
> This block is only available for **IVR** type applications. Refer to the Managing Applications page for more information.

You can use the **Transfer** block in the **Self Service** phase to transfer a call to another destination.

You can sequentially place multiple **Transfer** blocks with different settings, so that if a transfer fails in one block, your application proceeds to the next block. When a **Transfer** block successfully transfers the call, the application moves to the **Finalize** phase, ignoring any subsequent blocks in the **Self Service** phase.

Transfer tab

Click the **Destination** drop-down menu to select a target destination for the call.

Next, enter a timeout value for the application to wait for the transfer to proceed before moving to the next block.

Results tab

Select the variable that will store the result of this **Transfer** block execution.

Voice Mail Block

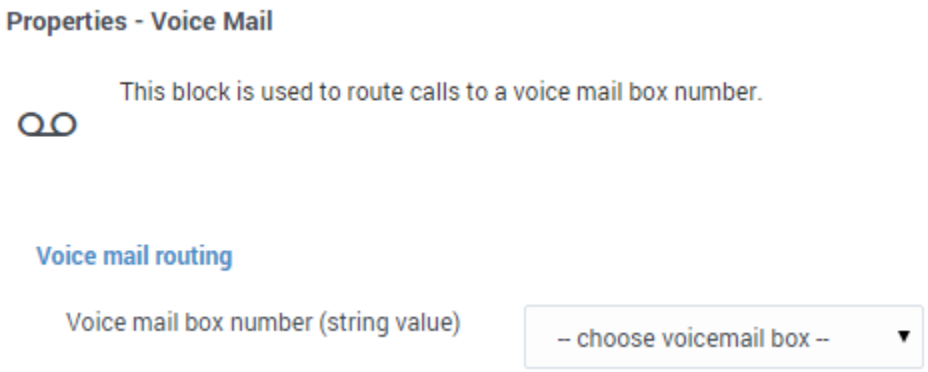
You can use the **Voice Mail** block in the **Assisted Service** phase to route calls to voicemail.

You can sequentially add multiple **Voice Mail** blocks with different settings, so that if routing fails, the next block is used. When a **Voice Mail** block successfully routes the call to voicemail, the application moves to the **Finalize** phase, ignoring any subsequent blocks in the **Assisted Service** phase.

Using this block

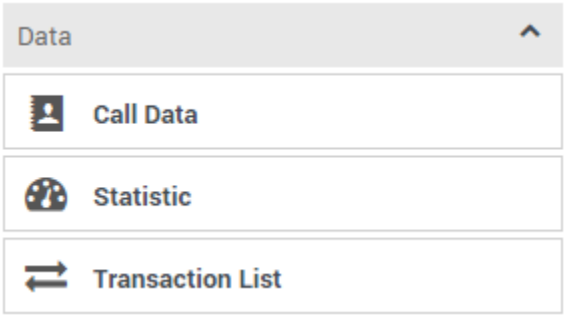
Voice mail routing

Select the voicemail number to which you want to route the call.



Data Blocks

These blocks are used for various data handling functions, such as retrieving information about a call that was created before the application started running, saving some information about a caller (or a call) to a database so that it can be processed later by another application, specifying keys to be read from Call Data and stored into application variables, or using statistics in the segmentation logic to route calls.



Use the links below to learn more about each block.

Call Data

Reads or writes data that can be accessed from both inside and outside an application.

Used in: **Initialization**, **Self Service**, **Assisted Service**

Statistic

Gets statistic values for queues or agent groups.

Used in: **Initialization**, **Assisted Service**

Transaction List

Assigns transaction list values to variables.

Used in: **Initialization**, **Assisted Service**

Call Data Block

Important

The **Call Data** block has the following restrictions when used in the **Self Service** phase:

- Non-variable key names must follow standard JavaScript rules for variable names, even if the keys are not variables. For example, key names must:
  - Not contain spaces.
  - Only contain alphanumeric characters.
  - Not use a restricted variable name (see the Restricted Variable Names section, below).
- Call data cannot be deleted. The **Edit Data** tab does not have the **Remove** option, as is available in the **Assisted Service** phase.

You can use the **Call Data** block to read and write data that can be accessed from both inside and outside the application.

The data format consists of key-value pairs (KVP). That is, for every key in the call data, you can associate a value with it. This value can be be an integer, character string, binary (boolean) type, or as a list of KVPs. (See an example of how to do this.)

About key-value pair lists: When a key-value pair list is used as a value in the user data for a **Call Data** block in the **Self Service** phase, the user data is attached to the call, but the platform encodes any special characters that appear in the user data, such as single quotation marks ('), double-quotation marks ("), backslashes (\), commas (,), colons (:), semicolons (;), and so on.

While data in application variables only exist within the application, information stored in Call Data can persist and be retrieved even after the application finishes running. This is useful for retrieving information about the call that was created before the application started running, or for saving some information about either a caller or a call to a database so that it can be processed later by another application.

Warning

The addition, updating, and removal of Call Data do not happen instantaneously while the application is running. That is, the application starts the operation to edit the Call Data, and then continues to the next block, without waiting for confirmation that the Call Data is successfully modified.

For this reason, Genesys recommends that you do not write a key-value pair and then attempt to read back the same key from the Call Data within an application. There is no guarantee that the write operation has finished, so a read operation could potentially give you either an old value or an updated value on different

runs through the application.

Instead, if you need to access a Call Data value that was already edited within the application, Genesys recommends that you use the corresponding application variable.

Important

- See the Restricted Variable Names section, below, for a list of variable names that must not be updated by the **Call Data** block.
- You must escape quote characters in values by using a preceding backslash. For example:
  - Incorrect: `Joe's Pizza`
  - Correct: `Joe\'s Pizza`
- The **Call Data** block only accepts string type keys and values, so you do not need to enclose these values in single quotes.

Read Data tab

Use the **Read Data** tab to specify keys to be read from Call Data and stored into application variables. You can toggle the key being read between a variable and a string.

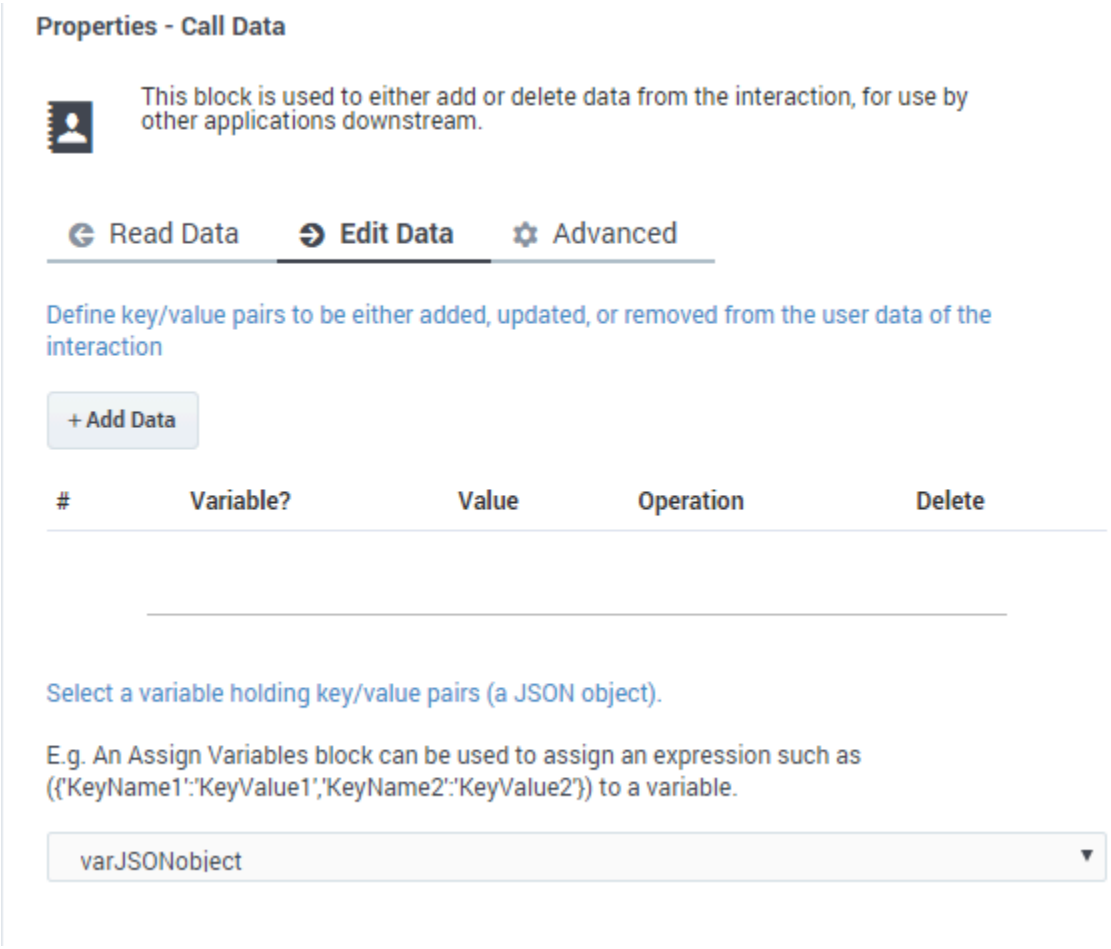
Edit Data tab

If this block is in the **Self Service** phase, use the **Edit Data** tab to specify key-value pairs to be written to the call data.

If this block is in the **Assisted Service** phase, use the **Edit Data** tab to specify key-value pairs to be written to the call data, either by adding data with new keys (**Add/Update** operation), updating data for existing keys (**Add/Update** operation), or deleting data for existing keys (**Remove** operation):

- **Add/Update** - Add or update data with the specified key. This operation automatically adds the key-value if the specified key does not yet exist in the Call Data, or, if the provided key does exist in the Call Data, it automatically updates data for the key. You can toggle both the key and the value independently between a variable and a string.
- **Remove** - Provide the key for data you want to delete, which you can toggle between a variable and a string. At runtime, if the key you try to delete does not exist, nothing happens.

You can also add a list of key-value pairs by selecting a variable that holds key-value pairs in a JavaScript object.



Advanced tab

If this block is in the **Assisted Service** phase, use the **Advanced** tab to add an extension as a key-value pair to the key. Click **Add Extension Data** to add an extension as a key-value pair to this block. The value type can be a string or integer.

If you want to use a variable for the **Key** or **Value**, select the **Variable** checkbox and then select a variable from the drop-down menu. If the **Value** is an integer, select the **Integer** checkbox.

> ### Important
> You do not need to enclose extension values in quotes. However, if the quote is part of the value, you must escape the quote character by using a preceding backslash. For

example:

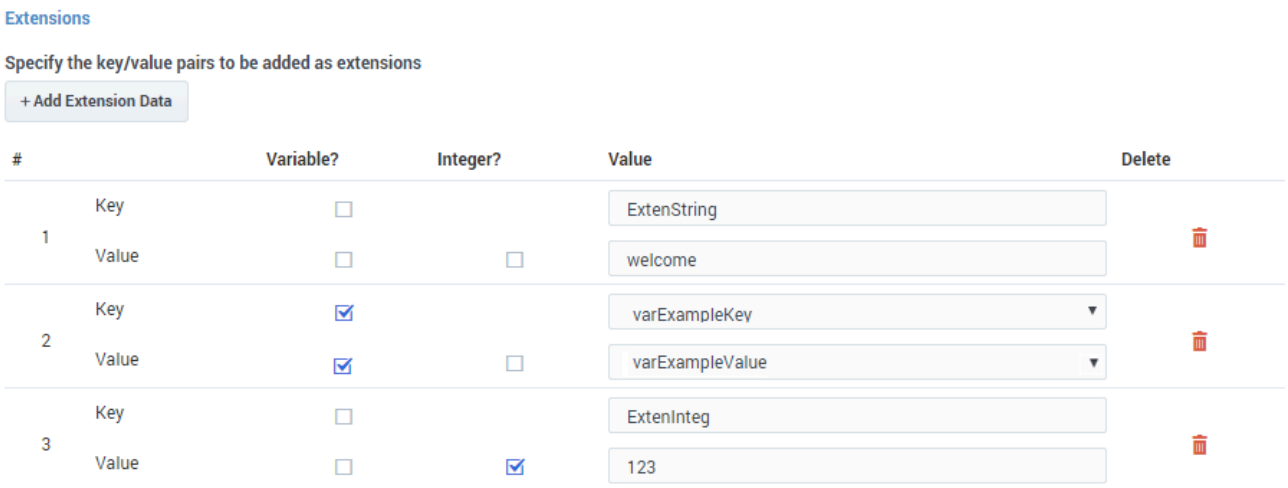
- Incorrect: `Joe's Pizza`
- Correct: `Joe\'s Pizza`

Important

Designer displays an error message if Extension Data is added, but the **Key** and **Value** settings are not defined.

Example

This example shows a few different ways that key-value pairs can be added as extensions:



Scenarios

If you want to:

- Read the key **userphoneno** into application variable **phoneno** that you have previously created:
  - In the **Read** section, click **Add Key**.
  - Disable the **Variable?** check box.
  - In the **Key** field, enter `userphoneno`.

    - In the **Store in Variable** drop-down menu, select **phoneno**.

- Add the key **segment** with the value from the application variable **custsegment** that you have previously created:
    - In the **Edit** section, click **Add Data**.
    - Click **Add/Update**.
    - Disable the **Variable?** check box for **Key**.
    - In the **Value** field for **Key**, enter `segment`.
    - Enable the **Variable?** check box for **Value**.
    - In the **Value** drop-down menu for **Value**, select **custsegment**.

Restricted Variable Names

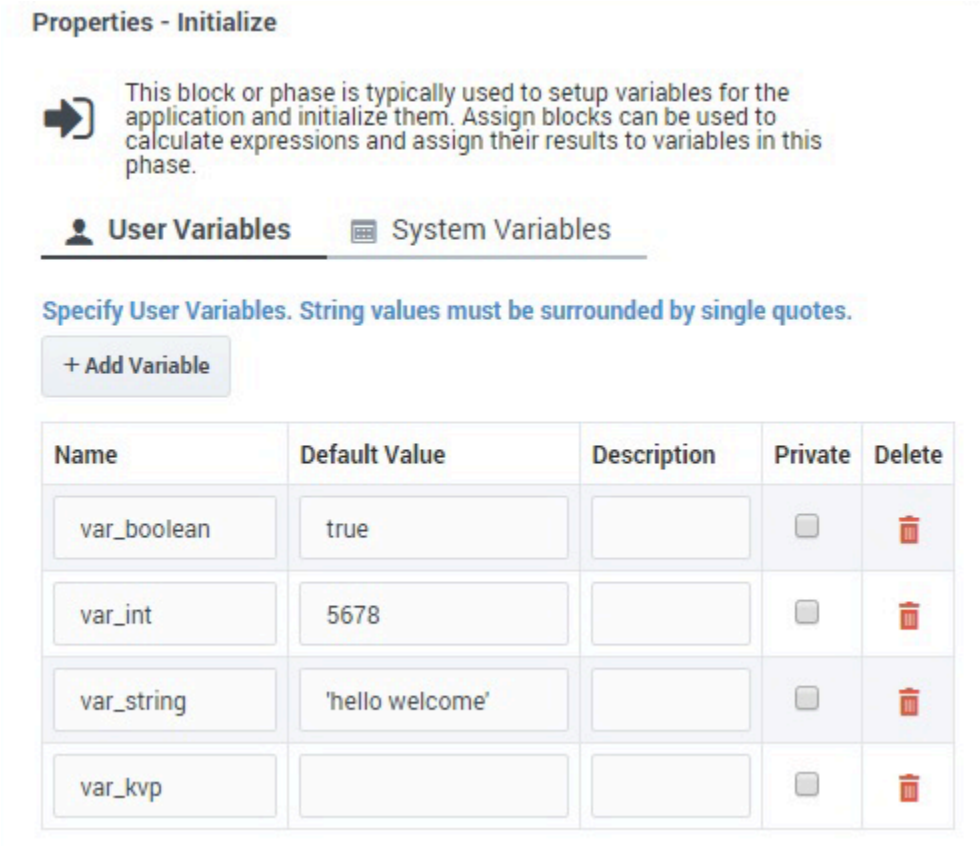
The following variable names are used by the Designer application and must not be updated by the **Call Data** block.

- **_CB_N_CUSTOMER_ABANDONED_WHILE_WAITING_FOR_AGENT**
- **_CB_SERVICE_ID**
- **_CB_T_CALLBACK_ACCEPTED**
- **_CB_T_CUSTOMER_CONNECTED**
- **_CB_T_SERVICE_START**
- **_COMPLETE_CS**
- **_SEND_FINAL_UE**
- **CustomerSegment**
- **EXECUTION_MODE**
- **GMS_UserData**
- **gvp-tenant-id**
- **gsw-ivr-profile-name**
- **GSYS_IVR**
- **GSYS_SystemApplicationDisposition**
- **IApplication**
- **IApplicationVersion**
- **IPurpose**
- **orsurl**
- **orssessionid**

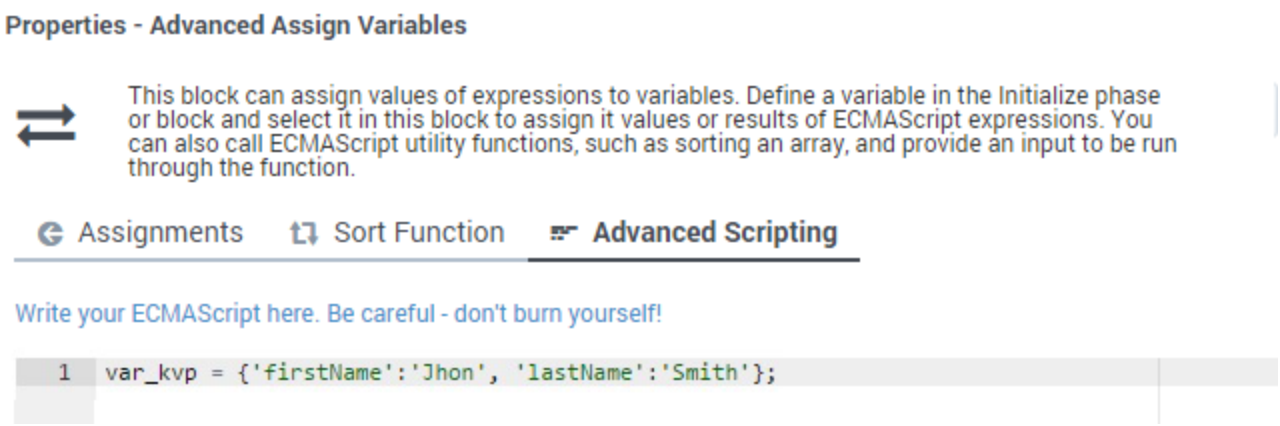
Example

This shows how you can assign the value of a key-value pair in the **Call Data** block to a variable in the **Assisted Service** phase.

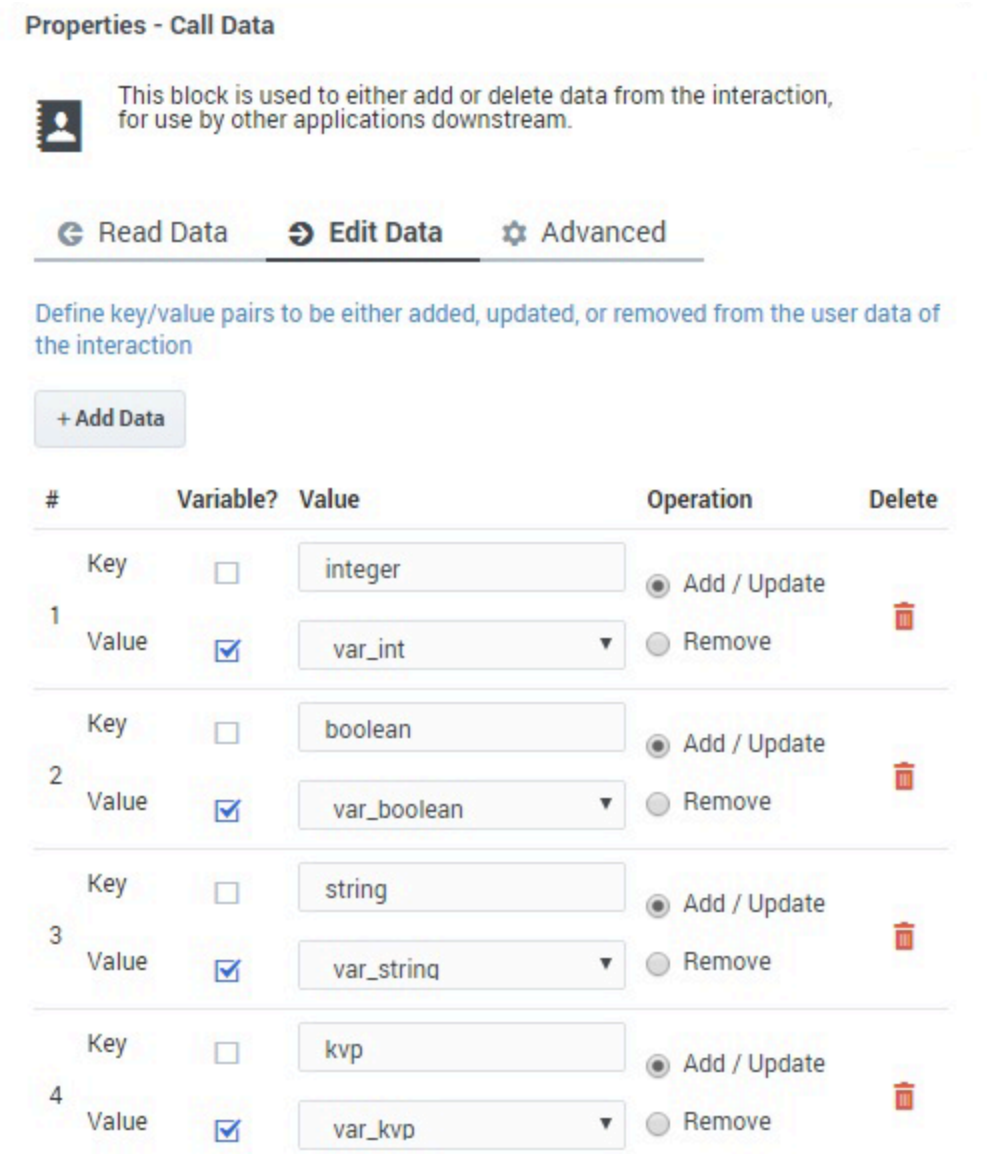
First, initialize your variables as various data types (integer, character string, binary [boolean], and so on), and create a variable for your KVP (in this example, we are using *var_kvp*):



Next, initialize a variable as a list of kv-pairs using the Assign Variables block. This example shows this using an ECMAScript on the **Advanced Scripting** tab:



Finally, add the user data as a kv-pair in the Call Data block:


Properties - Call Data
This block is used to either add or delete data from the interaction, for use by other applications downstream.
Read Data
Edit Data
Advanced
Define key/value pairs to be either added, updated, or removed from the user data of the interaction
+ Add Data
# Variable? Value Operation Delete
1 Key integer Add / Update
Value var_int Remove
2 Key boolean Add / Update
Value var_boolean Remove
3 Key string Add / Update
Value var_string Remove
4 Key kvp Add / Update
Value var_kvp Remove

Statistic Block

You can use the **Statistic** block in the **Assisted Service** phase to get the statistic value on queues or agent groups. These statistics can be used in the segmentation logic to route calls.

Using this Block

Click **Add Assignment** to add a statistic.

Select a variable in the **Variable** drop-down menu.

Click the **Type** drop-down menu to select the type of statistical value: current (**default**), **minimum**, **maximum**, or **average**.

Click the **Statistic** drop-down menu to select a statistic. For more information, see Statistic Types.

Click the **Object** drop-down menu to select the Agent Group or Queue for which the statistic is needed. When using a variable, the variable value must be a string in the form `ObjectName@.Type`, where the type can be one of `Q` (Queue) or `GA` (Agent Group), (for example, `Billing@.Q`).

Statistic Types

The following statistic types are available on the Statistic block:

Agent Loading

This statistic selects agents within an agent group and calculates a vector based on three values:

- The number of busy DNs.
- The agents' time in Ready state (the same value as the **Time In Ready State** statistic)
- A random number.

How It Works

The following assumes you are routing to multiple agents who are available in a group, and the routing engine must select one agent. The engine always selects an agent in the group according to **Agent Loading** statistics. It does this as follows:

- Evaluating the number of busy DNs and selecting the agent with the lowest number of busy DNs.
- If the number of busy DNs is equal among the agents, the routing engine selects the agent with the longest time in Ready state.
- If time in Ready state is equal among the agents, the routing engine uses a random number.

> **Important**
> The **Agent Loading** statistic works for agents only - it is not used for Agent Groups.

Agent Loading Media

This statistic is similar to **Agent Loading**, but it also considers other media types, such as chat and email channels.

The **Agent Loading Media** statistic ensures that the routing engine always distributes interactions evenly among agents.

Agent Occupancy

This statistic enables the routing engine to route interactions to the least occupied agent (the agent with the lowest occupancy rate). The occupancy rate is determined by the ratio between the time the agent has been busy since last login compared to the agent's total login time.

The **Agent Occupancy** statistic enables the routing engine to evaluate multiple available agents and select the least occupied agent, which balances the workload among available agents.

Agent Occupancy is defined and calculated when the agent logs in and can be used only in statistics that are applied for an agent. This statistic cannot be used with agent groups.

How it works

Consider the following scenario:

- After login, Agent 1 was on a call for five minutes and was in a Ready state for five minutes. His occupancy is 50 per cent (5 / (5+5)).
- After login, Agent 2 was on a call for one minute and was in a Ready state for two minutes. His occupancy is 33 per cent (1 / (1+2)).

When using the **Agent Occupancy** statistic, the routing engine distributes an incoming interaction to Agent 2, as he is the least occupied agent.

Agent State

This statistic provides the current status of agents.

How it works

You can use the **Agent State** statistic when you want to make a routing decision based on the status of your agents (for example, the number of agents logged out, ready, or not ready).

> **Important**
>
> To use this statistic, the application developer must first create and format the variable value for agent ID (for example, <agentid>@.A.)

Agents Available

This statistic provides the current number of agents in Ready State within one or more agent group at any given point of time.

How it works

You can use the **Agents Available** statistic when you are routing to multiple agent groups and you want to make a routing decision based on agent availability in each group.

Agents Busy

This statistic provides the number of busy agents in an agent group at any given point of time.

How it works

You can use the **Agents Busy** statistic when you are routing to multiple agent groups and you want to make a routing decision based on the number of busy agents in each group.

Agents in Queue Login

This statistic provides the current number of agents who are logged into a queue. This applies to agents logged into a virtual queue or an ACD queue.

How it works

You can use the **Agents in Queue Login** statistic when you are routing an interaction and you want make a routing decision based on the number of agents logged into one or more specific queues.

Agents in Queue Ready

This statistic provides the current number of agents who are logged into a queue and in Ready state. This applies to agents logged into a virtual queue or an ACD queue.

How it works

You can use the **Agents in Queue Ready** statistic when you are routing an interaction and you want to make a routing decision based on the number of agents who are logged into one or more specific queues and are also in Ready state.

Agents Total

This statistic provides the total number of agents who are logged into an agent group.

How it works

You can use the **Agents Total** statistic when you are routing an interaction and you want to make a routing decision based on the number of agents who are logged into a specific agent group.

> **Important**
>
> Capacity Rules might affect the number of available agents that is reported by the **Agents Total** statistic. For example, if an agent is already handling the maximum number of interactions, as defined by a Capacity Rule, then the **Agents Total** statistic does not include this agent.

Calls Answered

This statistic provides the total number of calls answered by an agent or agent group.

How it works

You can use the **Calls Answered** statistic when you are routing an interaction to multiple agent groups and you want to make a routing decision based on the total number of calls answered by a specific agent group and pick the agent group with the lower total.

Calls Completed

This statistic provides the total number of calls that have been completed by an agent or agent group. This includes all types of calls (inbound calls, outbound calls, internal calls, and so on).

How it works

You can use the **Calls Completed** statistic when you are routing an interaction to multiple agent groups and you want to make a routing decision based on the total number of calls that have been completed by a specific agent group and pick the agent group with the lower total.

Calls in Queue

This statistic provides the current number of calls that are waiting in a specific queue. This could be an ACD queue or a virtual queue.

How it works

You can use the **Calls in Queue** statistic when you are routing an interaction to multiple queues and you want to make a routing decision based on the total number of calls in a specific queue and pick the queue with the lower total.

Calls Waiting

This statistic provides the current number of interactions waiting for a specific target. It takes into account all the interactions that are waiting for the targeted skill level, both directly and through groups. This statistic applies to the following objects:

- Agent.
- Agent Group.
- Destination Label.
- Place.
- Place Group.
- Queue (virtual and ACD).
- Queue Group.
- Routing Point (virtual and ACD).

How it works

You can use the **Calls Waiting** statistic when you are routing an interaction to multiple targets and you want to make a routing decision based on the total number of calls waiting for a specific target and pick a target with the lower total.

Estimated Waiting Time

This statistic provides the estimated wait time for queued calls, based on the average talk time and average answering time. This statistic applies to queues such as ACD queues, virtual queues, and routing points.

> **Important**
> This statistic does not take into account calls that are in transition.

Stat Server calculates the **Estimated Waiting Time** statistic according to the formula below:

```
LB =

If (ALI= 0) /*no agents at all, wait forever*/ 10,000,000,000

Else if (AR <= CIQ) /*no available agents, use expected waiting time*/AHT * (CIQ - AR +1)/ALI

Else /*enough agents, waiting time is 0, use minus occupancy*/ – (AR - CIQ)/ALI

Where:
LB or LB(Q) = calculated StatLoadBalance
CIQ or CIQ(Q) = calls waiting in queue
AR or AR(Q) = number of ready agents logged into the queue
ALI or ALI(Q) = number of all agents logged into the queue
AHT or AHT(Q) = the average handling time for agents logged into the queue
```

Expected Waiting Time

Similar to the **Estimated Waiting Time** statistic, the **Expected Waiting Time** statistic also provides wait-in-queue estimates for the last interaction that entered a virtual queue. This statistic has been designed for the multimedia model but assumes agents cannot handle more than one simultaneous non-voice interaction at a time.

This statistic applies to all types of media and is calculated and refreshed internally for the last 600 seconds.

Load Balance

Load balancing between routing targets helps to ensure there is an equal distribution of interactions among queues, DNs, agent groups and queue groups. This statistic considers the number of calls distributed between objects, the percentage of busy agents, the expected waiting time, or other routing features.

When this statistic is defined, the routing engine automatically counts calls that are routed to different DNs and queues and adjusts the logic so there is an equal distribution of interactions across the specific DNs or queues.

Position in Queue

This statistic provides the exact position of an interaction in a queue.

Interactions can be moved based on their position in the queue. Interactions of high value can be moved to a different queue to provide a quicker response.

Service Factor

This statistic is a percentage/interval pair that specifies a certain percentage of interactions must be handled in a certain period of time (for example, 80 percent of interactions in 20 seconds). The statistic is applied against an agent group.

The routing engine calculates the **Service Factor** at an interval of every 50 interactions or 30 seconds. Based on this information, the routing engine decides whether to expand or contract the current set of agents available for the next interval.

The internal algorithm within the routing engine calculates and manages this scenario based on three values:

- SLReal = The percentage of calls distributed in *Y* seconds.
- SLWarn = The percentage of calls distributed in *3/4 * Y* seconds.
- AWT = The average waiting time for distributed calls.

The routing engine uses SLReal, SLWarn, and AWT to determine whether to adjust the current group of target agents as follows:

- If SLReal equals the **Service Factor** percentage, the routing engine makes no change to the current working agent group.

- If SLReal falls below the specified service factor (*SLReal < X*), the routing engine compares the current AWT with the AWT previously measured.
- If the current AWT is less than the previous AWT, the **Service Factor** is improving. The routing engine assumes that the service factor will continue improving and makes no change to the current working agent group.
- If the current AWT is greater than the previous AWT, the routing engine adds agents to the working agent group according to the formula *1/4*(N-M)*, where *N* equals the number of selected agents and *M* equals the ideal set of agents.
- If the **Service Factor** level is acceptable but SLWarn is less than the percentage that should be achieved in *3/4 * Y* seconds for *X* per cent that was set as the **Service Factor**, the routing engine initiates one of the following preventive actions:
  - If the current AWT is greater than or equal to the previous AWT, the routing engine adds agents to the working agent group according to the formula *1/8 * (N-M)*.
  - If the current AWT is less than the previous AWT, the routing engine tries to reduce the number of agents in the current working agent group according to the formula *1/8 * (N-M)* for only those agents who are not ideal.
  - If the service factor SLWarn is greater than or equal to *Y* and the current AWT is less than the previous AWT, the routing engine tries to reduce the number of agents in the current working agent group according to the formula *1/4 * (N-M)* for only those agents who are not ideal.

> **Important**
>
> Routing Platform cannot reduce the number of working agents if all agents are currently handling interactions.

Time in Ready State

This statistic provides the total current time of an agent in Ready state. It is calculated based on availability - that is, there is no activity currently in progress on a particular DN and the agent has placed the DN in Ready state. The **Time in Ready State** statistic aggregates the total time for the state and this information can be used to build the target list.

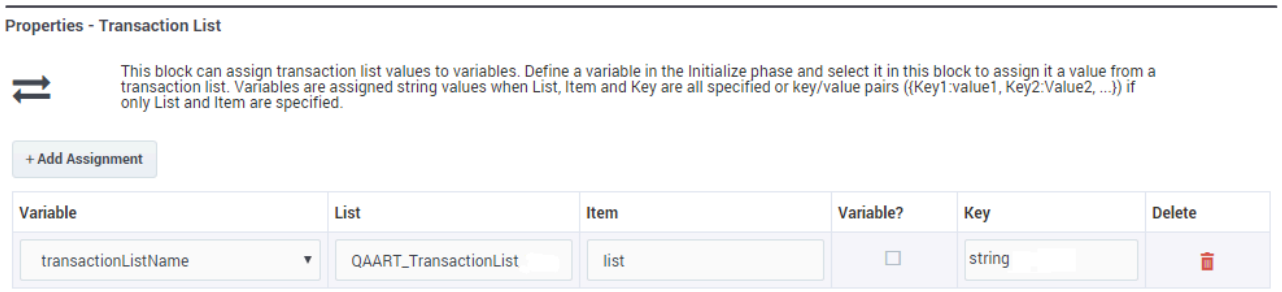
You can use the **Time in Ready State** statistic to route interactions to the agent that has been waiting in Ready state for the longest time.

Transaction List Block

You can use the **Transaction List** block to assign transaction list values to variables. You can define a variable in the **Initialize** phase and then select it in this block and assign it a value from the transaction list.

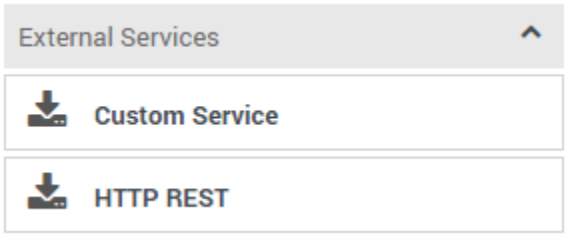
Variables are assigned string values when **List**, **Item**, and **Key** are all specified, or key-value pairs if you only specify **List** and **Item** (for example, `{Key1:value1, Key2:Value2}`).

Example



External Services Blocks

These blocks allow your application to interact with an external service, such as a custom service that Genesys has provided to your company, or an external system that stores and exposes data through a REST web service.



Use the links below to learn more about each block.

Custom Service

Provides inputs to a custom service that Genesys created for your company.

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

HTTP REST

Accesses external systems using RESTful API (over HTTP).

Used in: **Initialize**, **Self Service**, **Assisted Service**, **Finalize**

Custom Service Block

You can use the **Custom Service** block to access a custom service that was created by Genesys for your company.

You can provide input to the service. The resulting variable from the block is true if the service request is successful, otherwise the result is false. This result is available for use in later blocks.

Service Details tab

Select the service name and action to use in this **Custom Service** block.

Select **Disable DTMF buffering** if you want to prevent any DTMF inputs made during fetch audio playback from being buffered and carried forward into subsequent User Input or Menu blocks.

If you enable the **Play fetch audio** check box, you can specify an audio resource to play to the caller while the custom service is being fetched.

> **Important**
> Only Announcements containing audio files are supported. TTS audio will not be played.

- Enable the check box beside the **Play fetch audio** check box to specify a variable.
- In the **Play fetch audio minimum for** field, you can enter the minimum length of time to play the audio, even if the custom service has arrived in the meantime.
- In the **Start fetch audio after** field, you can enter a period of time to wait before audio is played.

> **Important**
> In the **Self Service** phase, fetch audio playback stops when the end of the audio file is reached, even if the service request is still in progress. In the **Assisted Service** phase, fetch audio playback loops until the service request times out.

Input Parameters

In the **Input Parameters** tab, specify the input expected by the custom service.

- **Name** - Specify the name of the parameter expected by the custom service.
- **Type** - The type of parameter (variable or literal).

- **Value** - Specify the parameter value to pass to the input.

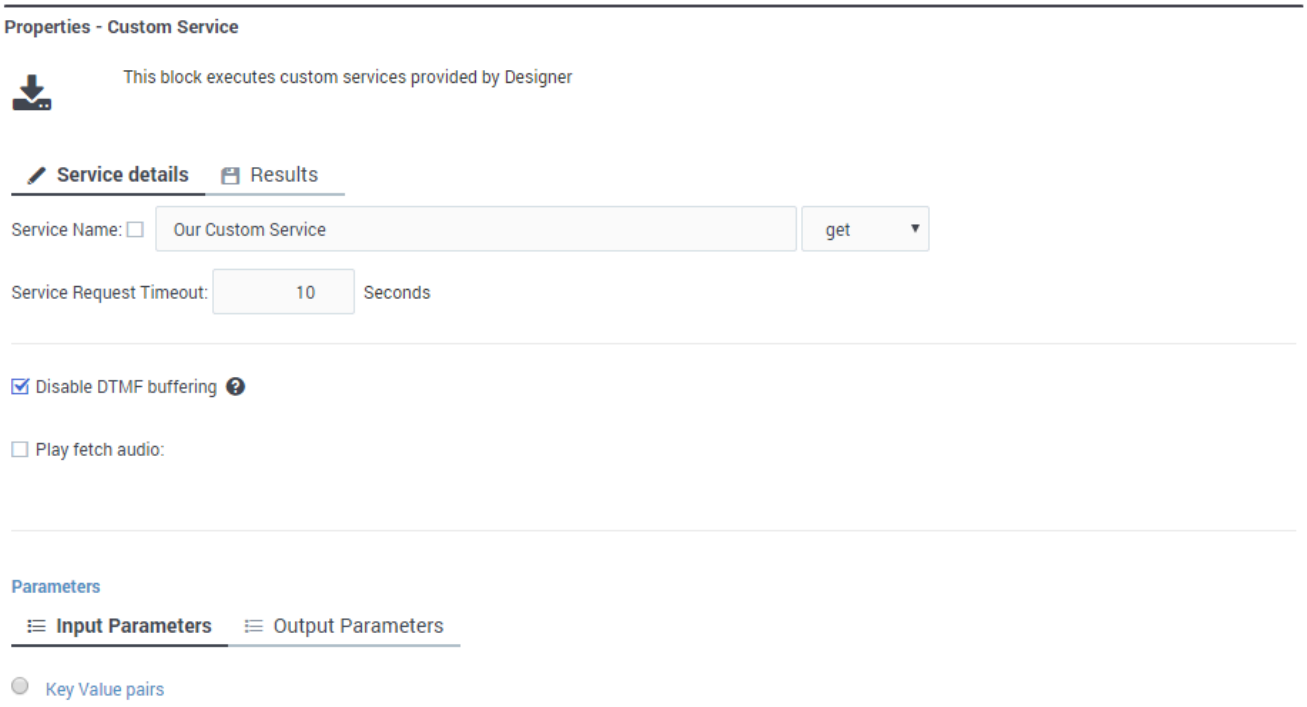
Output Parameters

In the **Output Parameters** tab, specify how and where to store the results of the custom service.

- **Variable Name** - Select the application variable in which to store the data.
- **JSON Expression** - Specify the key in which you expect the result to be in the response object. See the code sample and table below for an example.

```
{
        "thing": {
                "otherthing": "abc"
        },
        "arrayofthings": [
                "thing1", "thing2"
        ]
}
```

JSON Expression	Result
`thing.otherthing`	**abc**
`arrayofthings[1]`	**thing2**



Results tab

Select a variable to store the outcome status (**true** or **false**) of the Custom Service request.

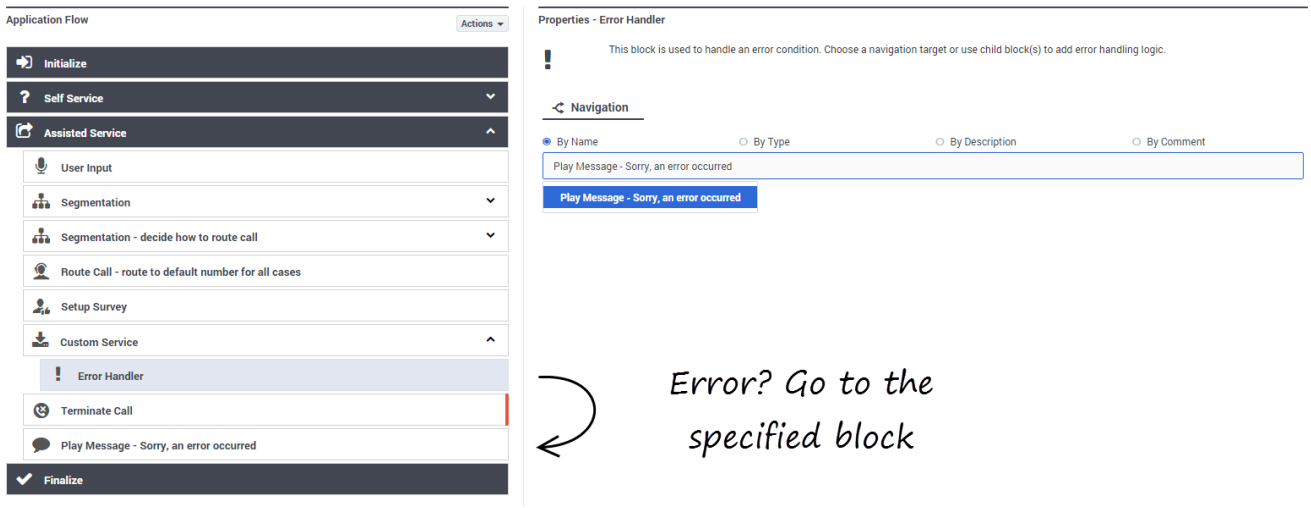
You must also select an action to take if the fetch operation is not successful. You can choose to "Continue with normal processing" or "Execute error handler blocks".

If you select "Execute error handler blocks", an **Error Handler** child block appears under the **Custom Service** block.

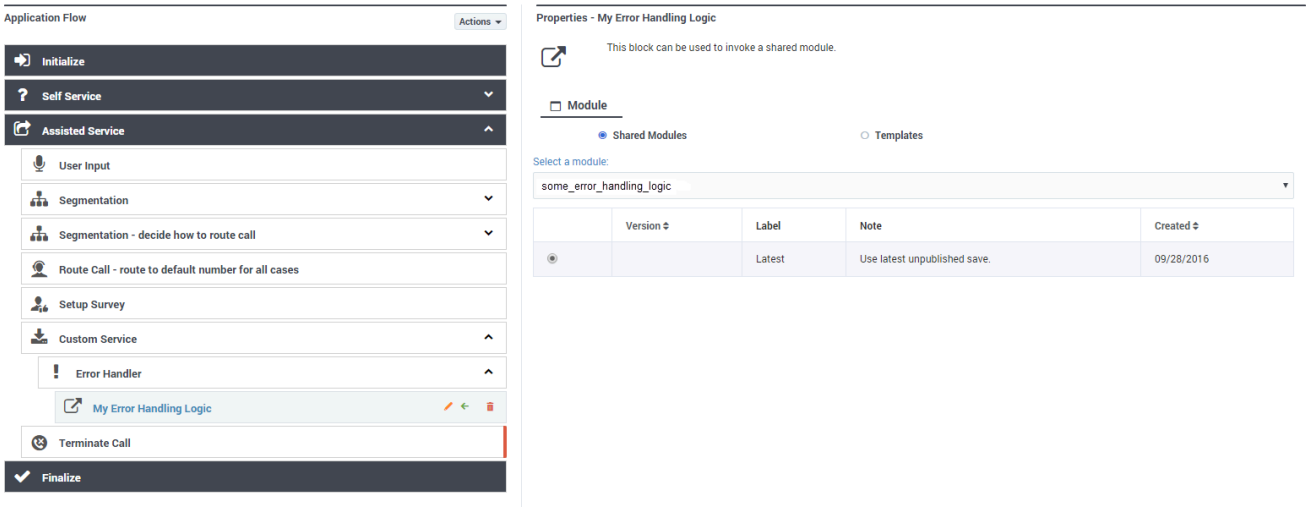
Use the **Error Handler** block to send the application to another target block that you select from the **Navigation** tab, or add child blocks that will perform the actual error handling.

Examples

In this example, the **Navigation** tab is used to specify a target block. If there is an error, the application will go to the **Play Message** block and play an error message:



In this example, a child block is used to invoke a module that will perform the error handling:



Tip

- If you select a target block from the **Navigation** tab, then any child blocks you've added to the **Error Handler** parent block are ignored.
- Standard validation rules still apply — any child blocks that you add to the **Error Handler** block must be valid for the application phase in which they are being used.

HTTP REST Block

You can use the **HTTP REST** block for accessing external systems using a RESTful API, over HTTP. You can read or write to these web services, although routing applications typically read from web services.

You can read or write to any external system that houses and exposes data through a REST web service. This could be a generic web service, such as one that returns the weather forecast for a specific location, or one that converts a monetary value from one currency to another. Or this could be a company's internal web service that fetches a customer's account details and billing history from the company's internal databases.

This block can be used in all four phases of the application.

Tip

- Check that the RESTful API you are accessing will return data in the format that you expect. While most web services typically return JSON data, there are some that may not. You may want to use an external tool to test the RESTful API outside of Designer to ensure it behaves the way you expect, before attempting to access it within your application.
- If the request timeout period is reached and no response is received from the REST web service, the output variables have a value of **undefined**.

Service Details tab

Enter the URL of the RESTful web service in the **HTTP URL** field. Enable the check box to use a variable, or disable the check box to use a string.

In the drop-down menu beside the **HTTP URL** field, select the HTTP method to access the web service: **get**, **post**, **put**, or **delete**.

If you are using **post** or **put** as the HTTP method, select an **Encoding Type**. (Otherwise, you will not see this option.)

In the **Request Timeout** field, enter the time, in seconds, that the application waits for a response from the web service before moving on to the next block.

Select **Disable DTMF buffering** if you want to prevent any DTMF inputs made during fetch audio playback from being buffered and carried forward into subsequent User Input or Menu blocks.

Select **Play fetch audio** if you want to specify an audio resource to play to the caller while the data is fetched.

> **Important**
>
> Only Announcements containing audio files are supported. TTS audio will not be played.

- Enable the check box beside the **Play fetch audio** check box to specify a variable.
- In the **Play fetch audio minimum for** field, you can enter the minimum length of time to play the audio, even if the document arrives in the meantime.
- In the **Start fetch audio after** field, you can enter a period of time to wait before audio is played.

> **Important**
>
> In the **Self Service** phase, fetch audio playback stops when the end of the audio file is reached, even if the fetch request is still in progress. In the **Assisted Service** phase, fetch audio playback loops until the request times out.

Input Parameters

In the **Input Parameters** tab, specify the inputs to the web service. You can choose either:

- **JSON Payload** — Send a JSON value from a variable as an input to the web service. This option is applicable only for **put** and **post** methods.
- **Key Value pairs** — Click **Add Parameters** and enter the **Name** of the parameter expected by the web service, and the **Value** to pass to the input. You can toggle the **Value** between a string and a variable.

Output Parameters

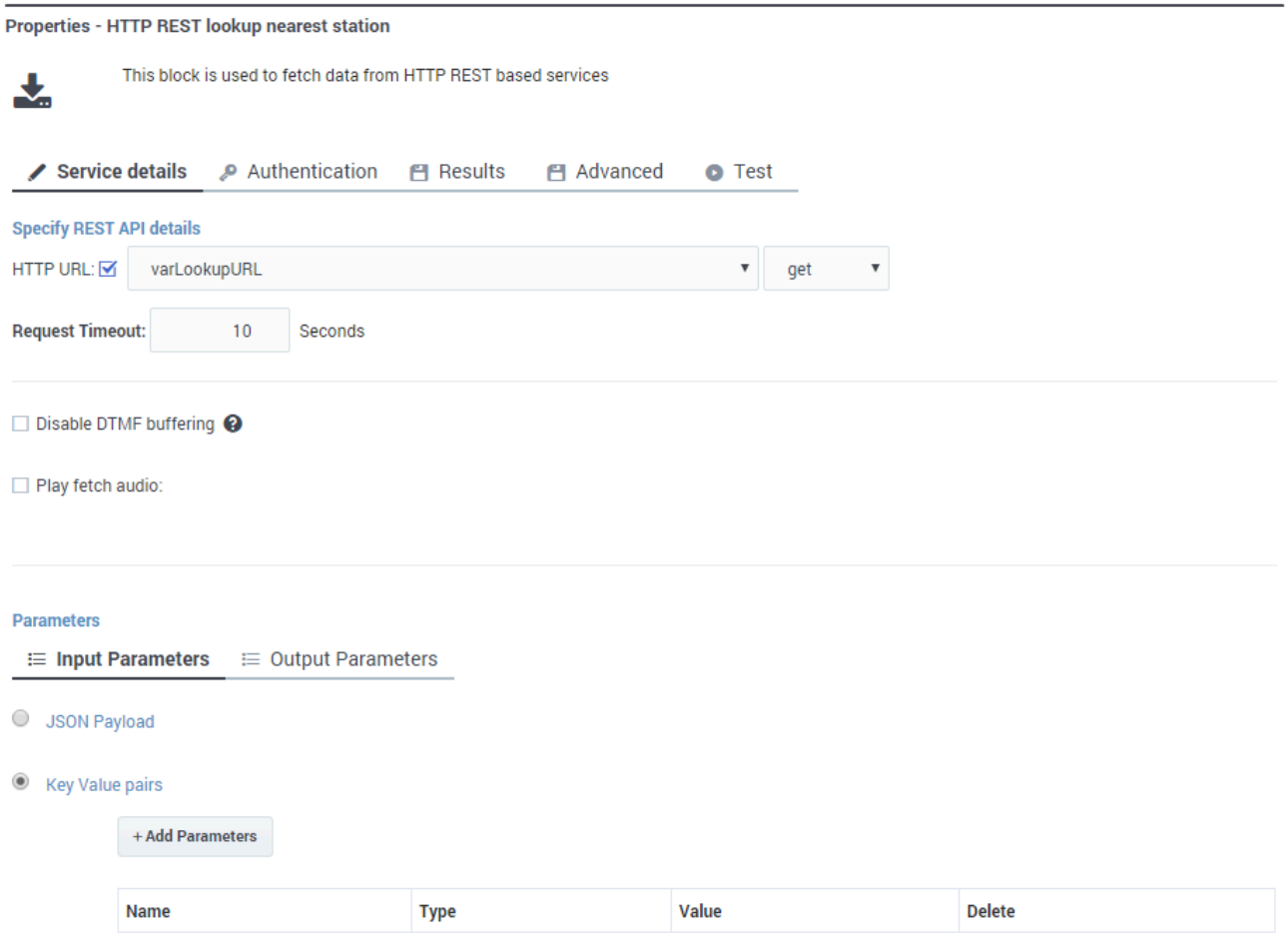
> **Important**
>
> You must only specify an output parameter if you are certain the web service will provide a consistent response. Otherwise, your application will generate an error if the web service provides a response that does not conform to what you have specified in the **JSON Expression** field (for example, a **400** error or a **200** code with no output). If the web service will not provide a consistent response, you can select a variable in the **Results** tab in which to store the entire HTTP response. Next, use an Assign Variables block to check for specific properties in the response and, if these properties are present, specify a JSON expression to assign to the variable.

In the **Output Parameters** tab, click **Add Parameters** to specify how and where to store the results of the web service call. The **Variable Name** is the application variable in which to store the data, and the **JSON Expression** is the key in which you expect the result to be in the response object.

See the code sample and table below for an example:

```
{
        "thing": {
                "otherthing": "abc"
        },
        "arrayofthings": [
                "thing1", "thing2"
        ]
}
```

JSON Expression	Result
thing.otherthing	**abc**
arrayofthings[1]	**thing2**



Authentication tab

Enable the **Enable Basic Authentication** check box to use HTTP basic authentication as part of the web service request. When enabled, the **User Name** and **Password** fields are displayed. Optionally, click the check box to select a variable for either of these fields.

Results tab

Select a variable to store the outcome status (**true** or **false**) of the HTTP fetch.

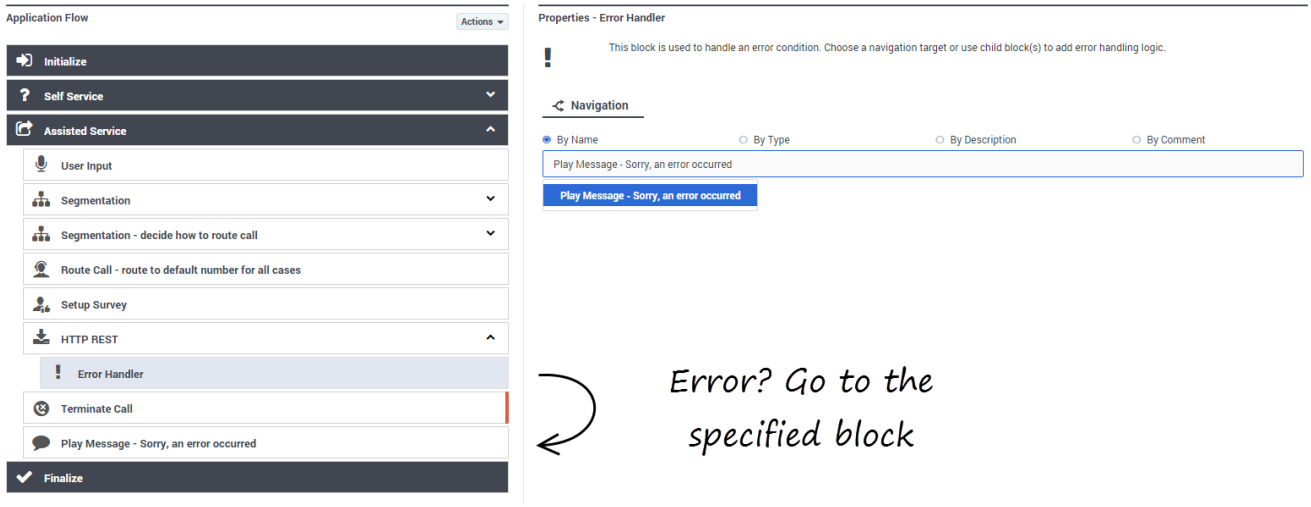
You can also select variables in which to store the data and headers of the HTTP response, and the HTTP error code if the operation failed.

You must also select an action to take if the fetch operation is not successful. You can choose to "Continue with normal processing" or "Execute error handler blocks".

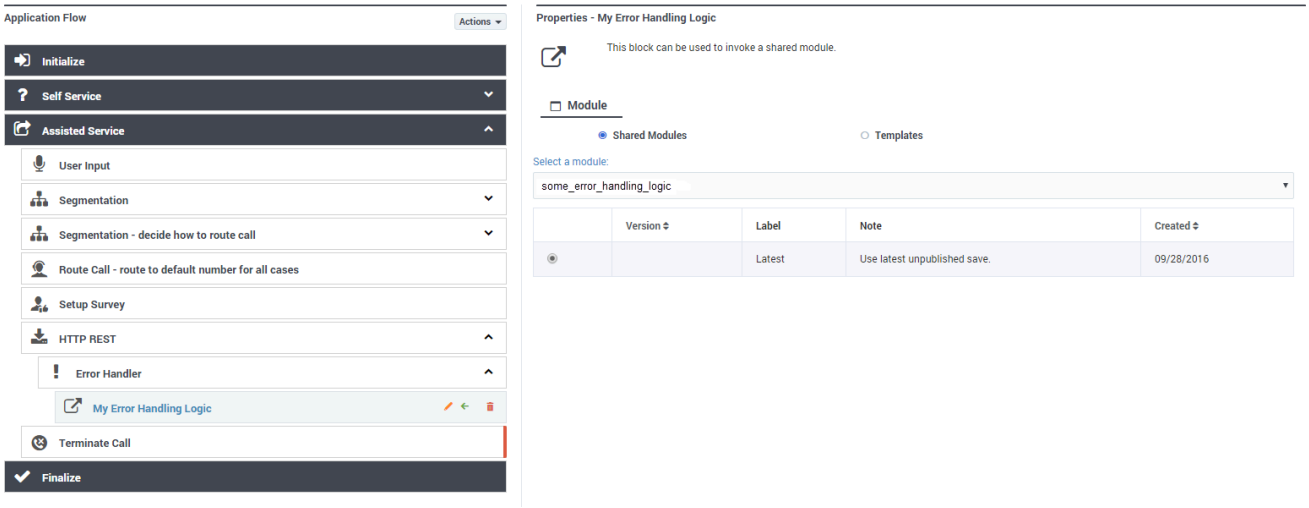
If you select "Execute error handler blocks", an **Error Handler** child block appears under the **HTTP REST** block.

Use the **Error Handler** block to send the application to another target block that you select from the **Navigation** tab, or add child blocks that will perform the actual error handling.

In this example, the **Navigation** tab is used to specify a target block. If there is an error, the application will go to the **Play Message** block and play an error message:



In this example, a child block is used to invoke a module that will perform the error handling:



Tip

- If you select a target block from the **Navigation** tab, then any child blocks you've added to the **Error Handler** parent block are ignored.
- Standard validation rules still apply — any child blocks that you add to the **Error Handler** block must be valid for the application phase in which they are being used.

Advanced tab

The **Use Designer service to make this request** check box is enabled by default. This allows the fetch request to use a HTTP proxy, which is typically required when sending requests to external resources.

Select **Internal Genesys Service** if the application is sending a fetch request to an internal Genesys service. This type of request does not go through a HTTP proxy.

Click **Add Header** if you want to use a custom HTTP header.

Important

If this **HTTP REST** block is used in a **Self Service** phase or shared module, there might be a processing delay due to the use of a proxy to perform the HTTP fetch. This delay does not apply to **HTTP REST** blocks that are used in **Assisted Service** phases or shared modules.

Test tab

The **Test** tab lets you test an API call from the block without making an actual test call.

Select the variables to be used as Input Parameters (make sure you specify them in the requested format, using single quotes for strings and "()" for JSON values) and any other variables to be used.

If the variables had a default value set in the **Initialize** phase, you can choose to keep those values or provide your own. The application will remember the values used the next time you open the application.

> **Important**
> Any literal values stored in the block will also be used for the test request.

Click **Send Test Request** to run the test and generate the results.

Scenarios

If you want to:

- Play weather information for a customer for whom you have a profile and address:
  - This scenario assumes that the weather API expects two input parameters (**date** and **location**) and provides its output in JSON format, under the key **result**. The corresponding input information is stored in two variables: **currentdate** and **zipcode**.
  - Add the **HTTP REST** block to the **Self Service** portion of the application, in a position after you have retrieved the customer location.
  - In the **HTTP URL** field, enter the URL of the weather web service (for example, `http://sample.webservice.com/api/weather/`).
  - Select **get** as the HTTP method.
  - In the **Input Parameters** tab, click **Add Parameters** twice.
  - For the first parameter, use the following information:
    - **Name**: `date`
    - **Type**: `variable`
    - **Value**: `currentdate`
  - For the second parameter, use the following information:
    - **Name**: `location`
    - **Type**: `variable`
    - **Value**: `zipcode`

- In the **Output Parameters** tab, click **Add Parameters** and use the following information:
  - **Variable Name**: `weather`
  - **JSON Expression**: `result`

Reporting Blocks

These blocks are used to manage certain reporting functions within an application, such as to start or stop an activity or to indicate certain points of an application's progress.

Activity

Starts or stops an activity.

Used in: **Self Service**, **Assisted Service**

Debug

Captures information used for debugging (Development stage only).

Used in: **Self Service**, **Assisted Service**

Milestone

Marks key moments within an application.

Used in: **Self Service**, **Assisted Service**

Activity Block

You can add the **Activity** block to Self Service or Assisted Service phases to start or stop activity in a report. You can also nest activities to provide additional details.

Do not use **Activity** blocks for modules, as Designer reports module activity automatically.

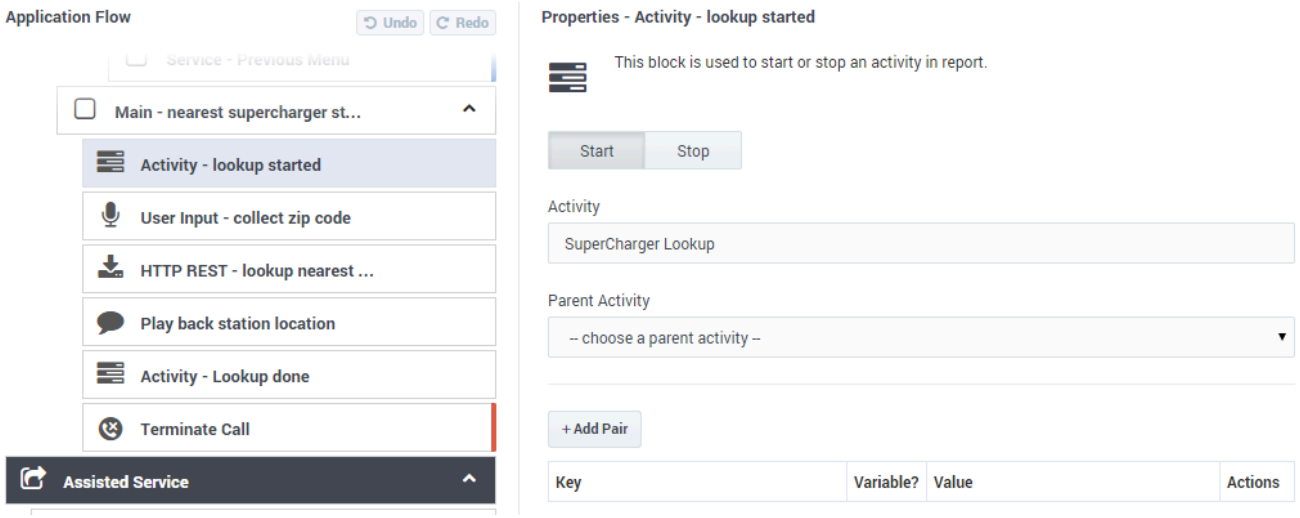
> **Important**
> VAR action IDs are stripped of spaces and pipe characters (|). This includes implicit actions that are generated when a caller enters a shared module.

Start tab

Click **Start** to indicate this block is the start of the activity.

Enter the name of the activity (IVR Action) in the **Activity** field. Optionally, you can select a parent activity by clicking the **Parent Activity** drop-down menu.

Click **Add Pair** to include data, values, or variables to store in the metric data of the activity.

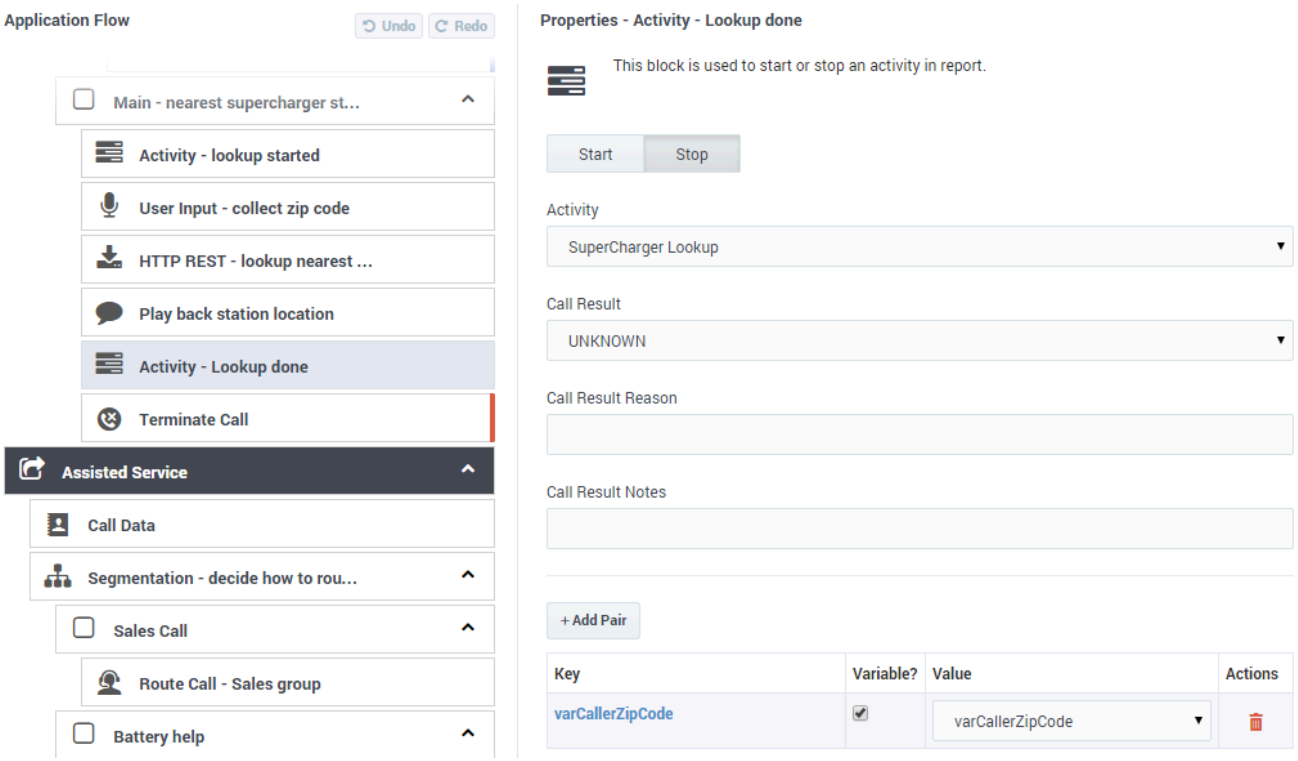


Stop Tab

Click **Stop** to indicate this block is the end of the activity.

Enter information in the following fields: **Call Result**, **Call Result Reason**, and **Call Result Notes**.

Next, click **Add Pair** to include data, values, or variables to store in the metric data of the activity.



Milestone Block

You can add the **Milestone** block to **Self Service** or **Assisted Service** phases to mark key moments while the application is running.

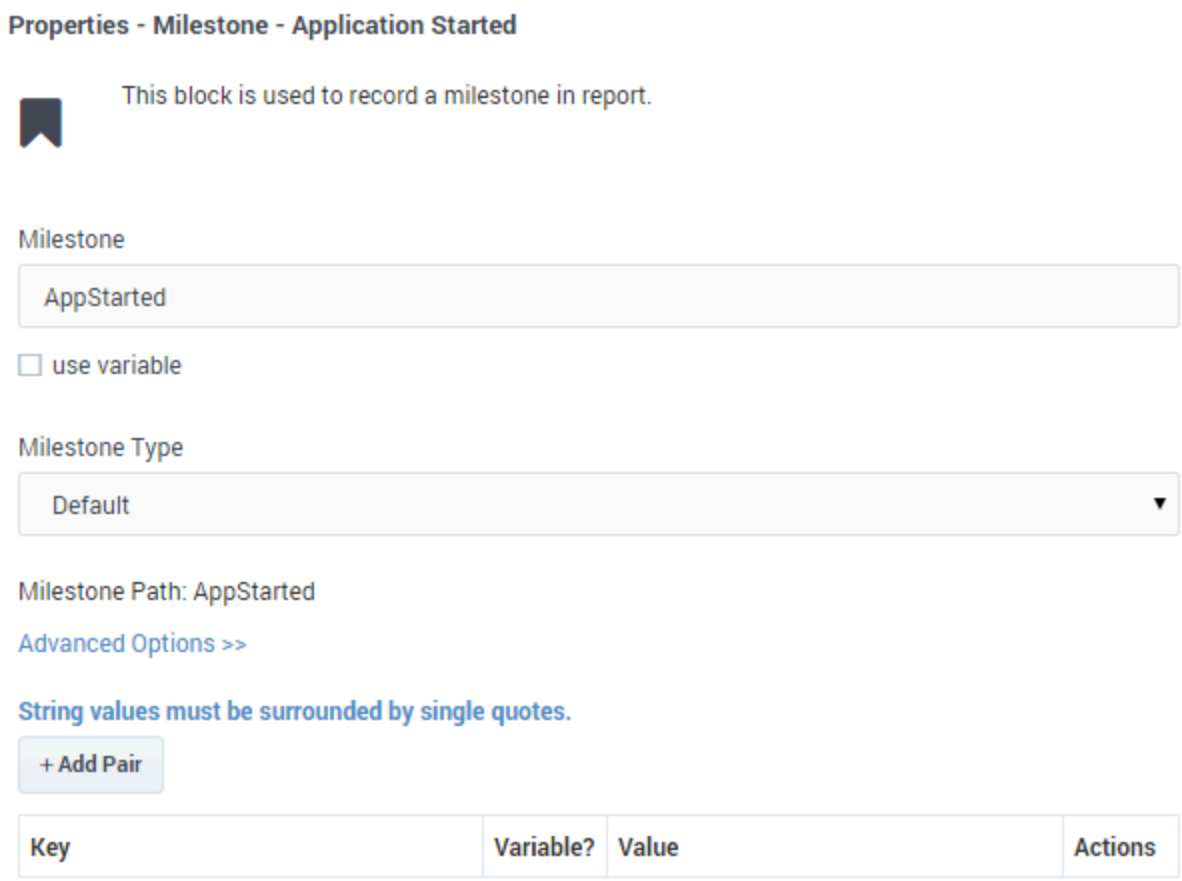
Using this Block

> **Important**
> VAR action IDs are stripped of spaces and pipe characters (|). This includes implicit actions that are generated when a caller enters a shared module.

Enter the name of the milestone in the **Milestone** field. Optionally, if this block is used in the **Self Service** phase, you can enable the **use variable** check box to use a variable for the milestone name.

In the **Milestone Type** menu, select **Default**. Only select **Survey** if this **Milestone** block will be used in a survey application (see the Survey section below for more information).

Optionally, enter additional information by clicking **Advanced Options** or **Add Pair**.



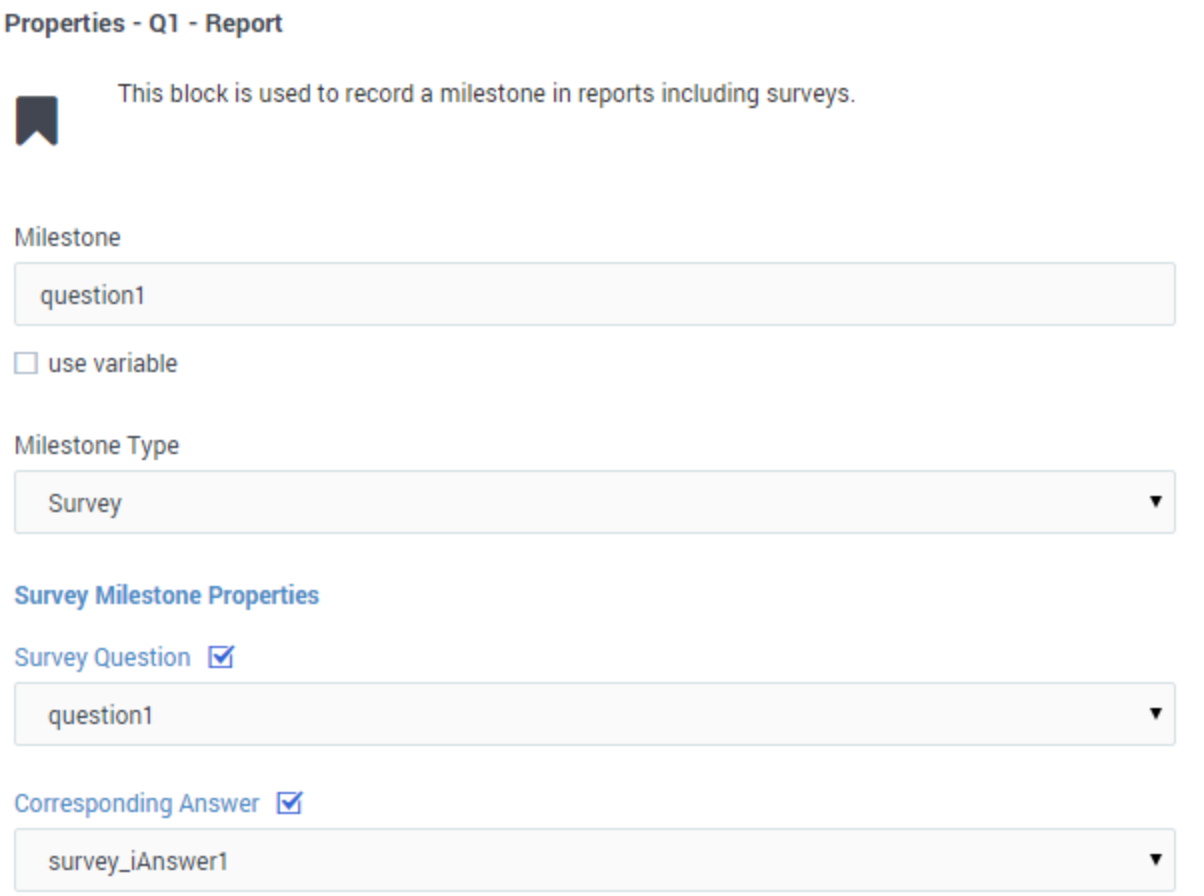
> **Important**
> When entering the payload **Key**, do not use single or double quotes.

Survey

The **Survey** type is reserved for survey applications.

Once you select **Survey**, the **Survey Milestone Properties** section appears. Configure the following:

- **Survey Question** - Select the variable that stored the survey question.
- **Corresponding Answer** - Select the variable that stored the survey answer.



Important

You can disable the check boxes if you would prefer to not use a variable for **Survey Question** and **Corresponding Answer**. However, Genesys recommends that you use variables for consistency and ease of use.

Debug Block

You can use this block during the application development phase to define a specific checkpoint in a module or application. This can be useful when you want to debug runtime logic that might not be running as intended.

For example, you might add some ECMAScript expressions to an Assign Variables block to assign values to certain variables, but discover that the logic isn't producing the desired result.

To assist in debugging this, you could add a **Debug** block immediately after the Assign Variables block to capture the values of the variables as they exist at that time. These values can then be reviewed in Designer Analytics, under the `debugcheckpoints` property of the Session Detail Record (SDR) .

> **Important**
>
> The **Debug** block is only processed if the application is in the Development phase. It is ignored if the application is in the Live stage.

Debug tab

Capture Checkpoint

Select this option to enable debugging for a specified checkpoint.

Checkpoint Name

Specify the name of the checkpoint to be captured. You can also specify a variable that holds the value of the checkpoint name.

Condition

Specify the condition as a script expression. For example:

```
var02 === 1
```

Data tab

This tab displays a list of variables that can be captured by the checkpoint. Select the variables that

you want to include.

Advanced tab

Enable the **Write these statements to platform logs** option if you want to write the results of the specified ECMAScript expressions to platform logs that can be reviewed by Genesys support. Click **Add Log** to enter the ECMAScripts.

You can also specify an error message statement to add to the logs if the expression evaluations result in an error.

Callback Block

> **Warning**
> This block is now deprecated. Designer now uses the Callback V2 block for this functionality. For more information about Callback, see the Callback blocks page.

> **Important**
> The **Callback** block relies on callback functionality provided by Genesys Mobile Services (GMS). Read the Callback User's Guide for more information on how to implement this feature.

You can use the **Callback** block in the **Assisted Service** phase of your **Default** type application for inbound calls. This block allows the caller to request a callback when the next agent is available (**Immediate** callback), or to schedule a callback for a more convenient time (**Scheduled** callback).

> **Tip**
> If you offer a **Scheduled** callback, you must also create a **Callback** type application to process the outbound call. See Scheduled Callbacks, below, for more information. To learn more about application types, see the Applications page.

Required Variables

The table below lists required user variables for the callback feature. You must create these variables before using the **Callback** block.

> **Important**
> You do not have to use the exact variable names suggested below. However, this page references these variable names throughout to provide clear and consistent instructions for most users.

Variable	Default Value	Description
offerImmediate	`true`	Stores whether **Immediate**

Variable	Default Value	Description
		callback is offered. Set the default value to `false` if you do not want to offer this callback type, or if you want to determine this value at runtime.
offerScheduled	`true`	Stores whether **Scheduled** callback is offered. Set the default value to `false` if you do not want to offer this callback type, or if you want to determine this value at runtime.
offerHold	`true`	Stores whether callers can decline the callback and remain on hold. Set the default value to `false` if you do not want to offer this callback type, or if you want to determine this value at runtime.
vq	Example: '*Callback_VQ*'	Specifies the Virtual Queue name to use for callback.
skill_expr	Example: '*Billing>0*'	Specifies a skill expression to use when routing callbacks.

Call Routing tab

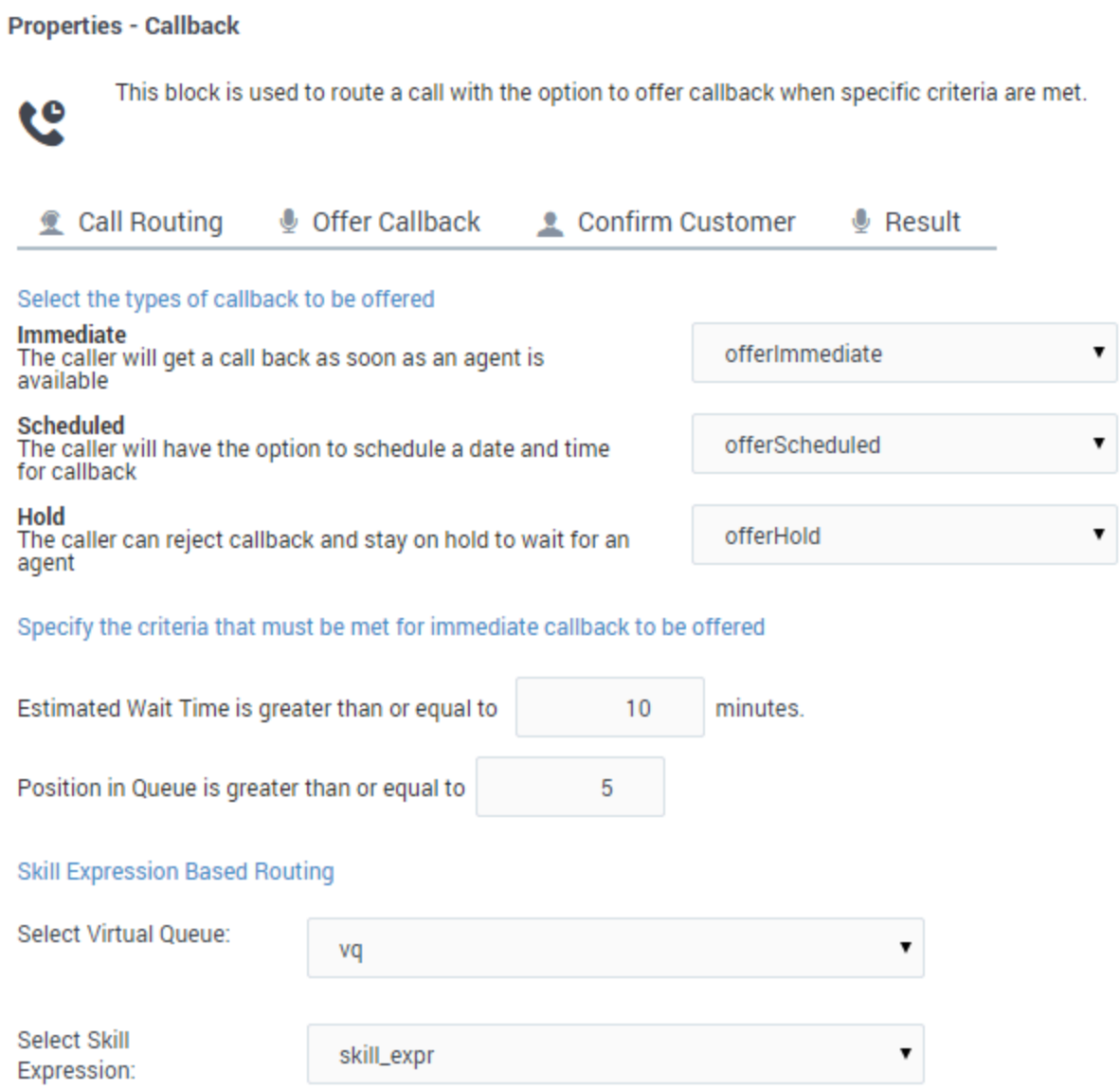
Specify the variables and rules to use when offering callback.

In the drop-down menus for **Immediate**, **Scheduled**, and **Hold**, select the variables that you created in the Required Variables section. See the figure below for an example.

If you are offering **Immediate** callback, you can also set the following options:

- **Estimated Wait Time is greater than or equal to** - Specify the minimum value for Estimated Wait Time, in minutes, before **Immediate** callback is offered. Set this value to `0` to always offer **Immediate** callback.
- **Position in Queue is greater than or equal to** - Specify the minimum queue position before **Immediate** callback is offered. Set this value to `0` to always offer **Immediate** callback.

In the **Skill Expression Based Routing** section, select the variables that you created in the Required Variables section. See the figure below for an example.



Offer Callback tab

In the drop-down menu, select **Template - Offer Callback** to use the pre-packaged template for callback.

The inbound callback feature is provided by a series of shared modules. The **Callback** block hands off the call to one main shared module that guides callers through the callback process. This shared module might rely on one or more supporting shared modules to extend its functionality (such as to collect a phone number or negotiate a time for **Scheduled** callback). When the callback process is complete, the main shared module returns the call to your application.

For ease of use, you can use shared module templates that provide pre-packaged callback functionality. The templates are read-only and cannot be edited or deleted. If you want to modify these templates, go to the **Shared Modules** list and click **Clone** beside a template to create a copy for editing.

Warning

Although you can copy a template to modify its prompts or behavior, you must not change its inputs or outputs. Doing so might cause unexpected behavior or application errors. If you want to change audio prompts only, you can modify audio resources in the **Callback** audio collection, which you can access by going to the Media Resources window.

Connect Customer tab

In the drop-down menu, select **Template - Calling Back** to use the pre-packaged template for callback.

Tip

You must also create an application of type **Callback** to provide the outbound call. See the Scheduled Callbacks section, below, for more information.

The outbound callback feature is provided by a shared module.

For ease of use, you can use a shared module template that provides pre-packaged callback functionality. The template is read-only and cannot be edited or deleted. If you want to modify this template, go to the **Shared Modules** list and click **Clone** beside the template to create a copy for editing.

Warning

Although you can copy a template to modify its prompts or behavior, you must not change its inputs or outputs. Doing so might cause unexpected behavior or application errors. If you want to change audio prompts only, you can modify audio resources in the **Callback** audio collection, which you can access by going to the Media Resources window.

Advanced tab

Greetings

Enable the check box beside **Customer Greeting** and/or **Agent Greeting** to play an audio file to that person while the call is being connected.

For customers, you might use this feature to play a legal disclaimer, or to announce that the call might be recorded (if you use call recording in your contact center). For agents, you might use a variable to announce the customer name or other relevant information.

After you enable **Customer Greeting** and/or **Agent Greeting**, you can select an audio file to play by clicking the icon in the **Announcement** field. This is useful for customer greetings that play a static disclaimer audio file.

Optionally, enable the **Var?** check box to use a variable to dynamically select the audio file. This is useful for agent greetings that use a variable to provide call-specific information, such as the customer name.

Result tab

In the drop-down, select a variable to store the outcome of the callback interaction. This step is optional.

Scheduled Callbacks

This section describes how to create a **Callback** type application to provide **Scheduled** callbacks. After you create your application, you must configure Genesys Mobile Services to use this application for outbound callback. Refer to the Callback User's Guide for more information.

In Designer, go to the Applications page and click **Add Application**. In the pop-up window, enter an application name and select **Callback** in the drop-down menu. Click **Create and Open**.

The **Callback** application type only has two phases - **Initialize** and **Finalize**. Drag a **Callback** block from the **Palette** and drop it in the **Initialize** phase.

Learn more about the tabs that are available for this block:

Confirm Customer tab

In the drop-down menu, select **Template - Calling Back** to use the pre-packaged template for callback.

The outbound callback feature is provided by a shared module.

For ease of use, you can use a shared module template that provides pre-packaged callback functionality. The template is read-only and cannot be edited or deleted. If you want to modify this template, go to the **Shared Modules** list and click **Clone** beside the template to create a copy for editing.

> **Warning**
> Although you can copy a template to modify its prompts or behavior, you must not change its inputs or outputs. Doing so might cause unexpected behavior or application errors. If you want to change audio prompts only, you can modify audio resources in the **Callback** audio collection, which you can access by going to the Media Resources window.

Result tab

In the drop-down, select a variable to store the outcome of the callback interaction. This step is optional.

User-Originated Callbacks

This section describes how to create a **Callback** application that provides **user-originated** callbacks. This allows a customer who is using a mobile device to request a callback and receive a push notification when an agent is available. The customer can then connect to the agent.

After you create your application, you must configure Genesys Mobile Services to use this application

for outbound callback. Refer to the Callback User's Guide for more information.

In Designer, go to the Applications page and click **Add Application**. In the pop-up window, enter an application name and select **Callback** in the drop-down menu. Click **Create and Open**.

You can only use this type of **Callback** application with the **Initialize** phase. Drag a **Callback** block from the **Palette** and drop it in the **Initialize** phase.

Learn more about the tabs that are available for this block:

Confirm Customer tab

Select **Customer dials in**. In a user-originated callback scenario, the customer initiates the call after receiving a push notification that an agent is available.

border|center

Advanced tab

Use the **Advanced** tab to configure options for **Greetings** and **Reporting**.

Configure a greeting message for the agent, customer, or both. The message plays just before the call between the agent and customer is connected and can be a variable or an announcement from the audio collection.

You can also choose to enable reporting on the reconnected call waiting for an agent by placing the call in virtual queue. Select a variable or existing virtual queue from the drop-down menu or append the callback virtual queue name with a suffix (for example, you can add *_out* to the name of the reconnecting virtual queue).

border|center

Result tab

(Optional) In the drop-down, select a variable to store the outcome of the callback interaction.

Valid values for user-originated callback:

- CALLBACK-ABANDONED_IN_QUEUE

- CALLBACK-ROUTED_TO_AGENT
- CALLBACK-PUSH_FAILED

Known Issues

- You cannot use callback if you also use priority increments for routing in the **Route Call** block. Callback requires all calls to keep their priority from the time they enter the queue. Calls can have different priorities that are set at the beginning of the call, but their priorities cannot be modified once they enter the queue.

Callback Blocks

These blocks manage options, rules, and features for Callback.

> ## Warning
> **Use templates or modules — not both.**
>
> Genesys recommends that you avoid mixing templates and modules in callback applications. When planning your applications, decide whether you are going to use templates *or* modules, and then be consistent with your choice.
>
> If you need to make changes to a template, clone *all* of the callback templates into corresponding modules, and then use those modules in your applications.

Use the links below to learn more about each block.

Callback V2

Offers callback and reconnects to the customer when an agent is ready.

Book ASAP Callback V2

Books an ASAP ("as soon as possible") Callback on Genesys Mobile Services (GMS).

Book Scheduled Callback V2

Books a scheduled Callback on Genesys Mobile Services (GMS).

Callback Availability V2

Retrieves the scheduled callback availability from Genesys Mobile Services (GMS).

Cancel Callback V2

Cancels an existing callback.

Check for Existing Callback V2

Checks if the customer's phone number already has an existing callback scheduled or queued in a particular Callback service in Genesys Mobile Services (GMS).

Validate Phone Number

Provides phone number validation and international phone number support for Callback V2.

Callback VQ Watermark

Checks the number of active callbacks that are currently queued for a specific virtual queue (VQ).

Callback V2

> **Important**
>
> The **Callback V2** block relies on callback functionality provided by Genesys Mobile Services. Read the Callback User's Guide for more information on how to implement this feature.

You can use the **Callback V2** block in the **Assisted Service** phase of your **Default** type application for inbound calls. Designer supports the following types of callbacks:

- Immediate (or in-queue) callbacks, where the caller requests a callback when the next agent is available.
- Scheduled callbacks, when the caller selects a preferred time and date for the callback.
- Web-invoked callbacks, where the caller requests a callback using an HTTP request (such as a website or a mobile application to request a callback when an agent is available).

You can use the **Callback V2** block in the **Initialize** phase of your **Callback** type application for Scheduled callback. This block processes the Scheduled callback at the desired time.

Before using the **Callback V2** block, you must first create a variable for the callback virtual queue. Then, you can use Special Days, Business Hours, and Data Tables (under Business Controls) to specify your business requirements and associate those settings with the virtual queue.

The settings for callback virtual queues are stored in the Callback_Settings data table.

> **Warning**
>
> The Callback V2 feature can only be used with Callback V2 blocks.

Sample callback scenarios

The following scenarios describe sample call flows for immediate and scheduled callbacks.

Immediate callbacks

- The session starts when the customer's call arrives.
- The caller is offered immediate (or in-queue) callback. They accept, and confirm the number they want to be called at.

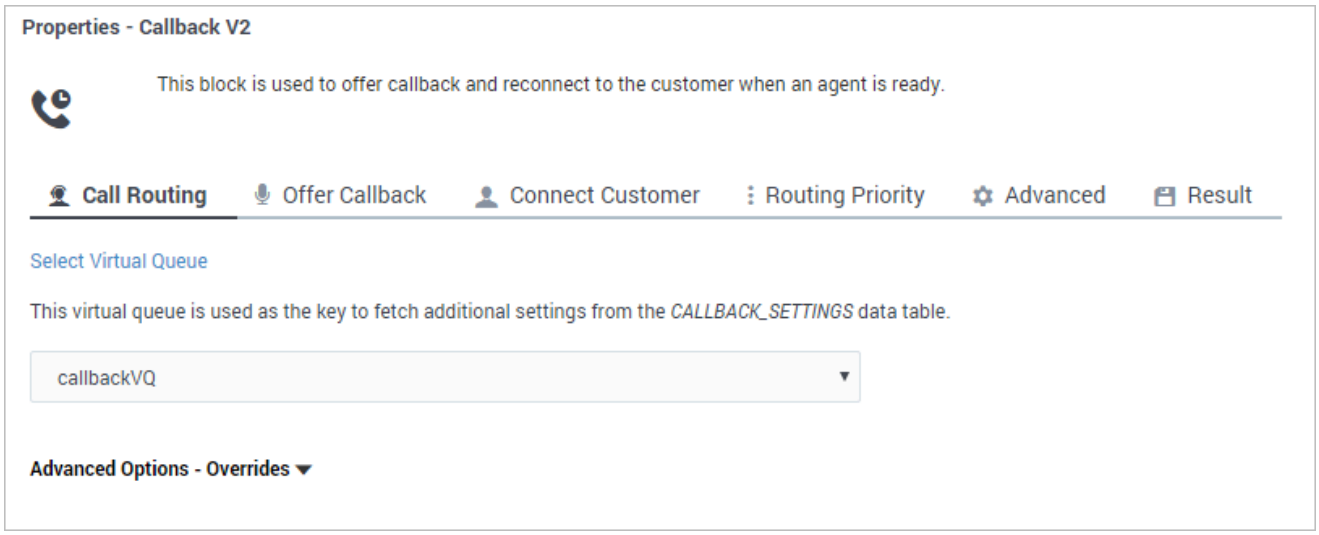
- At this point, the caller can hang up. The voice interaction is converted to a virtual call and added to the queue.
- While the virtual call waits in the queue, the session remains active and continues to monitor statistics for the call, such as the Estimated Wait Time (EWT) and it's position in the queue.
- When an agent that satisfies the required skill expression is ready, the customer is called.
- Music on hold plays while the call is being routed to the agent.
- Once the agent connects to the call, the virtual call is removed from the queue and the session ends.
- (Optional) If survey is enabled and the caller has agreed to take it, the caller is taken to the survey application after the agent disconnects.

Scheduled callbacks

- The session starts when the customer's call arrives.
- The caller is offered a scheduled callback. They accept, and confirm the number they want to be called at, along with the date and time when they would like to receive the callback.
- At this point, the caller can hang up.
- When an agent that satisfies the required skill expression is available, the customer is called.
- Music on hold plays while the call is being routed to the agent.
- Once the agent connects to the call, the virtual call is removed from the queue and the session ends.
- (Optional) If survey is enabled and the caller has agreed to take it, the caller is taken to the survey application after the agent disconnects.

Call Routing tab

Select the **Virtual Queue** that you are going to use for callback. Designer uses this Virtual Queue to fetch the associated configuration settings from the **CALLBACK_SETTINGS** data table.



Advanced Options - Overrides

(Optional) You can expand the **Advanced Options - Overrides** section to select your own variables for certain parameters. For example, your business might require that an offer or skill expression parameter override the current setting in **CALLBACK_SETTINGS** with a different value.

> **Important**
>
> Variables used for overrides must be provided as *boolean* values (for example, `true`/`false`, or `0`/`1`). Otherwise, Designer interprets the variable lookup as `false`.

file:des_callback_callrouting_var.png

Offer Callback tab

In the drop-down menu, select **Callback V2 - Offer Callback** to use the pre-packaged template for callback.

The inbound callback feature is provided by a series of shared modules. The **Callback V2** block hands off the call to one main shared module that guides callers through the callback process. This shared module might rely on one or more supporting shared modules to extend its functionality (such as to collect a phone number or negotiate a time for **Scheduled** callback). When the callback process is complete, the main shared module returns the call to your application.

For ease of use, you can use shared module templates that provide pre-packaged callback functionality. The templates are read-only and cannot be edited or deleted. If you want to modify these templates, go to the **Shared Modules** list and click **Clone** beside a template to create a copy for editing.

> **Warning**
> Although you can copy a template to modify its prompts or behavior, you must not change its inputs or outputs. Doing so might cause unexpected behavior or application errors. If you want to change audio prompts only, you can modify audio resources in the **Callback V2 Audio** audio collection, which you can access by going to the Media Resources window.

Connect Customer tab

In the drop-down menu, select **Callback V2 - Calling Back** to use the pre-packaged template for callback.

The outbound callback feature is provided by a shared module.

For ease of use, you can use a shared module template that provides pre-packaged callback functionality. The template is read-only and cannot be edited or deleted. If you want to modify this template, go to the **Shared Modules** list and click **Clone** beside the template to create a copy for editing.

> **Warning**
> Although you can copy a template to modify its prompts or behavior, you must not change its inputs or outputs. Doing so might cause unexpected behavior or application errors. If you want to change audio prompts only, you can modify audio resources in the **Callback V2 Audio** audio collection, which you can access by going to the Media Resources window.

Routing Priority tab

Enable the **Use Priority during Routing** check box to use priority-based routing, which prioritizes your calls depending on your business requirements.

To prioritize calls, you should set the **Initial Priority** based on your business segmentation (for example, *Gold* customers start at Initial Priority = 50, *Silver* customers start at Initial Priority = 30, and *Bronze* customers start at Initial Priority = 0).

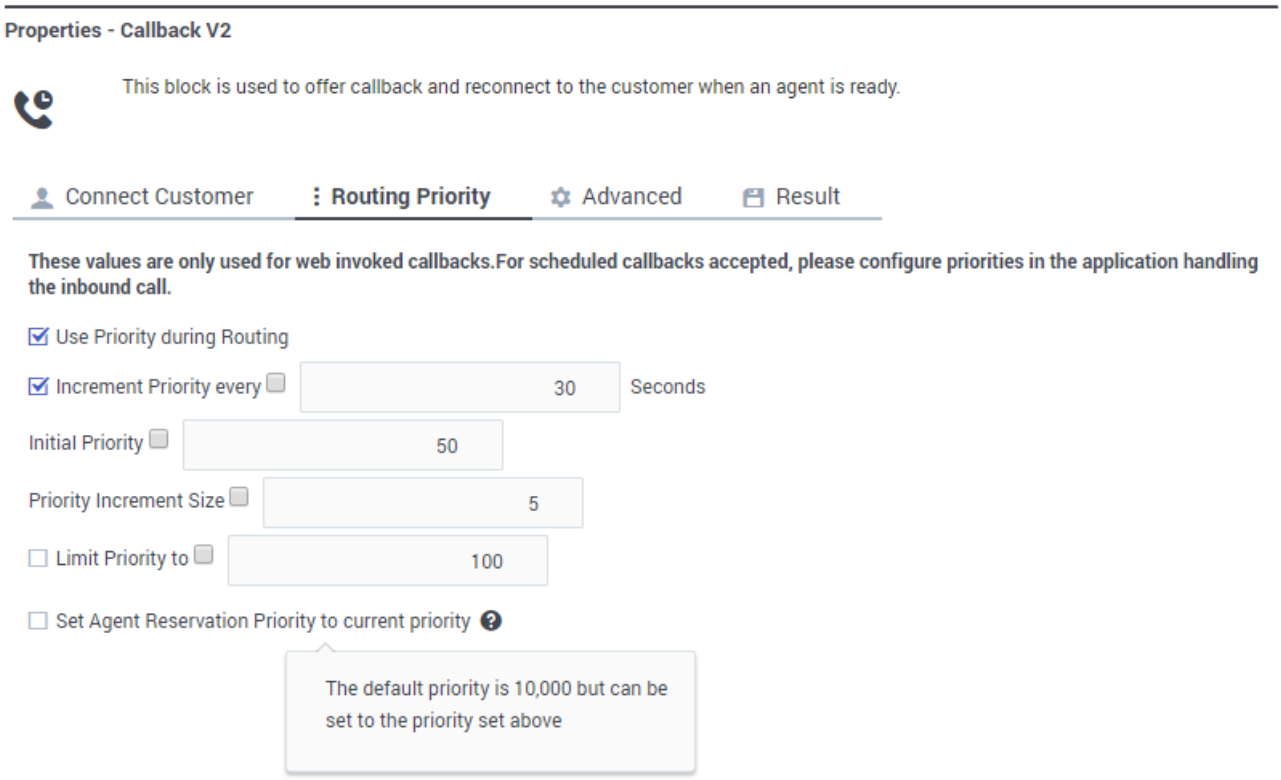
Enable the **Increment Priority every _____ seconds** check box to specify the time interval between priority increments. If you enable the other check box beside the field, you can select a variable that specifies the overall **Routing Timeout** and **Priority Increment Interval** properties.

If the **Increment Priority every _____ seconds** option is enabled, you can use the **Limit Priority to** option to set a maximum priority value. For example, if the initial priority is 50, you can use this option to not let the priority value increase beyond 100.

If you enable the other check box beside the field, you can select a variable for this option.

If the **Use Priority during Routing** option is enabled, you can choose to select the **Set Agent Reservation Priority to current priority** option. This applies the current priority (the priority of the call at the time an agent was found for the callback) to reserve the agent. If you do not select this option, the default value of 10,000 is applied to the request.

Example



Tip

Ideally, the Route Call block and **Callback V2** block should have their priorities synchronized, so that their rate of increase is the same. One way you can do this is by using variables for the **Initial Priority**, **Increment Priority every...**, **Priority Increment Size**, and the **Limit Priority to...** settings.

Advanced tab

Greetings

Enable the check box beside **Customer Greeting** and/or **Agent Greeting** to play an audio file to that person while the call is being connected.

For customers, you might use this feature to play a legal disclaimer, or to announce that the call might be recorded (if you use call recording in your contact center). For agents, you might use a variable to announce the customer name or other relevant information.

After you enable **Customer Greeting** and/or **Agent Greeting**, you can select an audio file to play by clicking the icon in the **Announcement** field. This is useful for customer greetings that play a static disclaimer audio file.

Optionally, enable the **Var?** check box to use a variable to dynamically select the audio file. This is useful for agent greetings that use a variable to provide call-specific information, such as the customer name.

Music on Hold

Enable **Music on hold** to select the music file that plays while callers are on hold.

Reporting

Enable **Put (re)connected call into a virtual queue** if you would like to place the real interaction in a separate virtual queue for reporting purposes, to differentiate between regular calls and calls routed to agents as a result of a callback.

You can either select a virtual queue that is defined in the application or you can add a suffix to the virtual queue that is used for the inbound call. For example, if the callback virtual queue is named *VQ1_cb*, and the suffix is *_out*, the reconnected virtual queue should be configured as *VQ1_cb_out*.

The following options enable you to specify the metrics to display in reporting:

Enable **Show the EWT of the inbound VQ when callback was offered** to specify the name of the inbound virtual queue. You can select a variable or one of the virtual queues available in the drop-down menu.

Enable **Show the threshold that was used to determine if callback should be offered** to specify the **Callback EWT Threshold value** (in seconds). You can select a variable or enter an integer.

Survey

(If survey is enabled) Enable **Route to a different RP than previously specified in Setup Survey block** if you need to change the routing point to use for the survey application after the agent disconnects. Otherwise, the routing point configured in the Setup Survey block is used.

Example

800px

Result tab

(Optional) In the drop-down, select a variable to store the outcome of the callback interaction.

Callback Settings Data Table

The callback settings for each virtual queue are stored in a special data table called **CALLBACK_SETTINGS**. You can view the settings for this data table by selecting it on the Data Tables page.

The data table includes a default queue that is already populated with the recommended values. To add a new virtual queue, simply add a new row to the data table. Each parameter is automatically assigned the default setting, but you can edit the values to further refine and customize the callback settings for each virtual queue.

If you are making changes to this data table, note the following:

- Genesys recommends that you do not set the *Callback_TTL* ("Time to Live") value lower than the default setting of `259200` seconds (3 days). This value specifies how long the callback service will be kept active in the system. If you set this value too low, the callback is removed from the system before the customer receives their callback. (This value does not apply to scheduled callbacks, as those can be booked up to a week in advance.)
- If your application is not automatically detecting the caller's number (ANI), you might have to use the *Dial Prefix* setting to enter a country calling code, or use the audio prompts to ask callers to include their country code when manually entering their callback phone number.
- The value for *Slot Duration (minutes)* must be a divisor of 60. The recommended values are 15 (default), `20`, `30`, or `60`, with `60` being the maximum value you can use.

Parameters

This data table contains the following parameters:

Setting	Description
VQ	Name of the Virtual Queue.
Immediate Enabled	Enables (or disables) the option to offer Immediate Callback.
Hold Enabled	Enables (or disables) the option to Hold (or Reject) a callback.
Logged In Check	Checks to see if any agents are logged in before offering Immediate Callback. If this feature is enabled and no agents are logged in, Immediate callback is not offered.
Immediate Blackout (minutes)	This value acts as a cut-off time (in minutes) before the end of the business day when Immediate callbacks won't be offered. For example, if the business closes at 5:00 PM and the **Immediate Blackout** value is set to 60 minutes (default), customers who call and receive an estimated waiting time that exceeds 4:00 PM (i.e. 60 minutes before closing) won't be offered an Immediate callback. Important If **Immediate Offer Hours** is configured, this option is ignored.
Callback Purge Time (minutes)	Duration (in minutes) to keep a callback session alive before we make a courtesy call to reschedule or cancel the callback because no agents were found and the callback cannot be processed. However, the courtesy call will be made at the end of the business day if the business closes before the callback session alive time. Tip You might want the courtesy outbound call to happen at the end of the business day. To do this, set the value of this parameter to a number that is greater than the total number of minutes the office is open. For example, an office with business hours of 09:00–17:00 would be open 480 minutes (or 8 hours). Setting this parameter to a higher value, such as 500, would initiate an outbound call to the customer after business hours (for example, to inform them that no agents were available and to provide them with the option to reschedule or cancel the callback).
Call Display Name	Name to display for Caller-ID.
Call Display Number	Number to display for Caller-ID.
Enable CPD	Enables (or disables) Call Progress Detection.

Setting	Description
CPD Timeout (seconds)	Specifies the maximum time (in seconds) allowed for Call Progress Detection after the call is connected.
Dial Prefix	The prefix to add to the phone number for outbound dialing. (This should only be used to add the country code, if desired. The + should not be added here, since it should already be configured in the dial plan.) **Important** If your application is not automatically detecting the caller's number (ANI), you might have to use this setting to enter a country calling code, or use the audio prompts to ask callers to include their country code when manually entering their callback phone number.
Dial Retry Timeout (seconds)	Time to wait (in seconds) before making another attempt to dial an outbound call, if the previous attempt failed.
Max Dial Attempts	Maximum number of times to try dialing an outbound call.
Min Time Before Callback (seconds)	Minimum time (in seconds) between the disconnection of the inbound call and the dialing of the outbound call.
Snooze Duration (minutes)	Time to wait (in minutes) before dialing a caller who chose the "snooze" option from the menu.
Push retry Timeout (seconds)	Time to wait (in seconds) before the next push notification is sent to the caller. **Important** This setting is currently not supported.
Max Push Attempts	The total number of push notifications to be sent to the caller. **Important** This setting is currently not supported.
Skill Expression	The skill expression to use for targeting an agent. (Example: `Billing>0&Collections>0`)
Attach Userdata	Specifies the format in which the user data should be attached to the interaction before it is routed to an agent.

Setting	Description
	• Selecting **single_json** will attach all user data as one JSON object (key: GMS_UserData). • Selecting **separate_keys** will attach each user data as a separate key. (The name of the key will be the same as the user data key.)
Business Hours	Name of the Business Hours entry for this VQ. This name must correspond to one of the entries in Business Hours. • If **Immediate Offer Hours** is not configured, these hours apply to both immediate and scheduled callbacks. • If **Immediate Offer Hours** is configured, these hours apply only to scheduled callbacks. • For immediate callbacks, these hours indicate when immediate callback is to be offered (up until the time specified by **Immediate Blackout**, if configured). • For scheduled callbacks, these hours indicate when timeslots will be available for booking scheduled callbacks.
Immediate Offer Hours	Name of the Business Hours object that defines the hours when Immediate callback is to be offered. This name must correspond to one of the entries in Business Hours. • Immediate callback will not be offered if the current time plus the Estimated Wait Time (EWT) is outside of the defined Immediate Offer Hours. For example, a customer calls your contact center on Monday at 16:55. If Immediate Offer Hours are defined as Monday-Friday, 09:00-17:00, and the EWT is 10 minutes, Immediate callback is not offered to the caller. Important This option disables the **Immediate Blackout** option.

Setting	Description
Call Direction	Determines who will initiate the call to the target.
Slot Capacity	How many callbacks can be offered for each slot.
Slot Duration (minutes)	Duration (in minutes) of the time slots for scheduled callbacks. **Important** This value must be a divisor of 60. The recommended values are 15 (default), 20, 30, or 60, with 60 being the maximum value you can use.
Routing Point	Routing Point (RP) to use for making the outbound call.
Callback Application	The name of the Designer callback application.
Callback TTL (seconds)	Specifies how long (in seconds) the callback service will be kept active in the system. **Important** Genesys recommends that you do not set this value lower than the default setting of 259200 seconds (3 days).

Book ASAP Callback V2

> **Important**
>
> - This block relies on callback functionality provided by Genesys Mobile Engagement (formerly known as Genesys Mobile Services). Read the Callback User's Guide for more information on how to implement this feature.
> - You must use this block in connection with the callback feature. See the Callback V2 block page for more information.

Use this block to book an immediate callback ("as soon as possible").

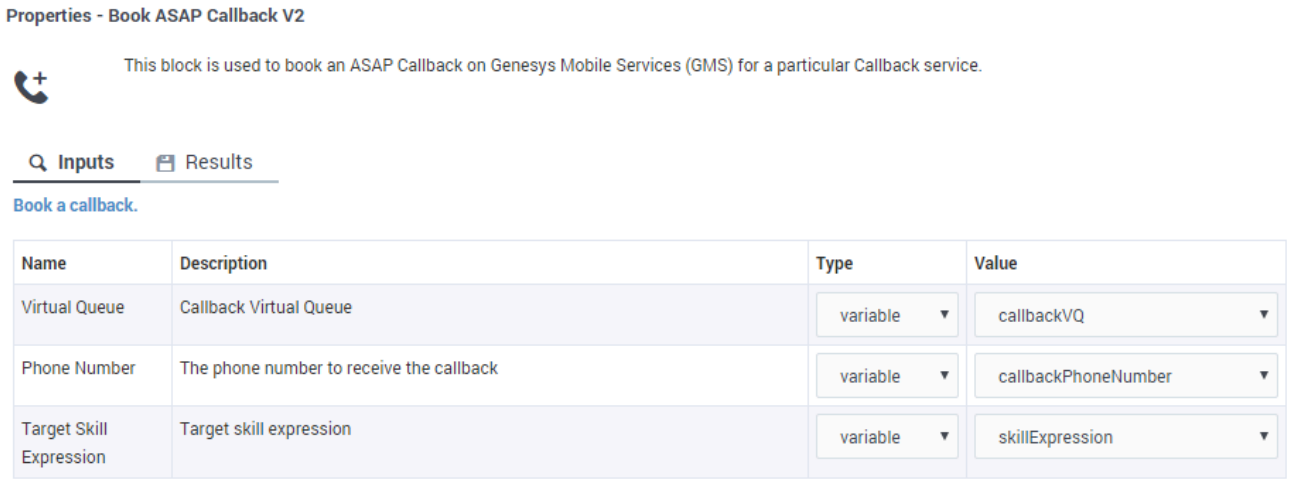
Inputs tab

Select the input **Type** and **Value** for the following parameters (this step is mandatory):

- Virtual Queue
- Phone Number
- Target Skill Expression

You can use literal or variable value types.

Example



Results tab

Select the variables that will store the results of the **Outcome** and **Callback ID** queries.

> **Tip**
> Genesys recommends that you use the system variable *GmsCallbackServiceID* to store the value of the **Callback ID**.

Example

Book Scheduled Callback V2

> **Important**
>
> - You must use this block in connection with the callback feature. See the Callback V2 block page for more information.

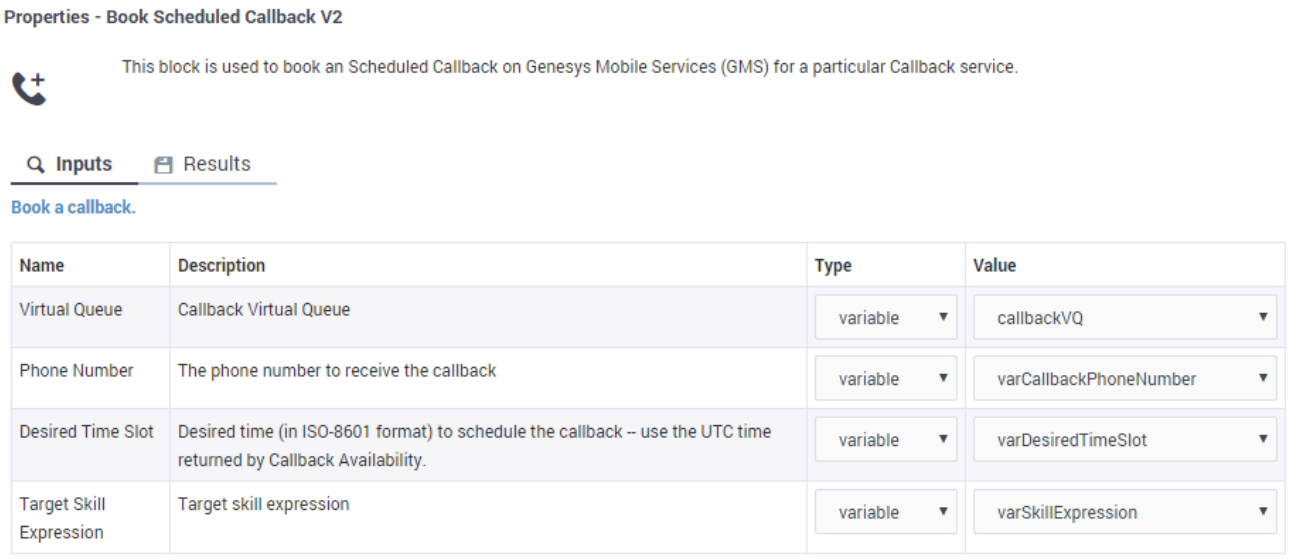
Use this block to book a scheduled callback.

Inputs tab

Select the input **Type** and **Value** for the following parameters (this step is mandatory):

- Virtual Queue
- Phone Number
- Desired Time Slot (this must be in ISO-8601 format, i.e. YYYY-MM-DD)
- Target Skill Expression

Example

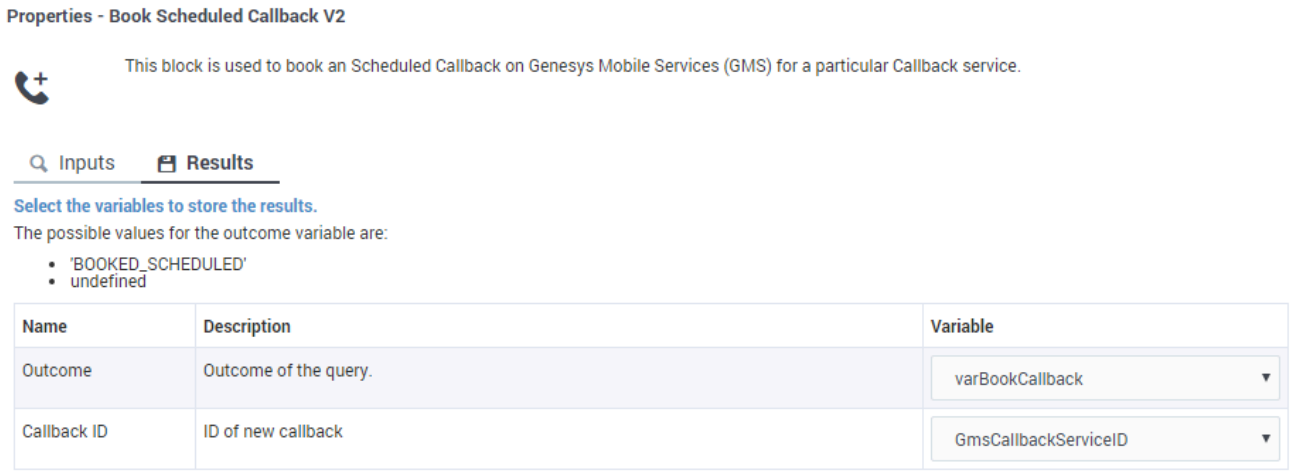


Results tab

Select the variables that will store the results of the **Outcome** and **Callback ID** queries.

> ### Tip
> Genesys recommends that you use the system variable *GmsCallbackServiceID* to store the value of the **Callback ID**.

Example



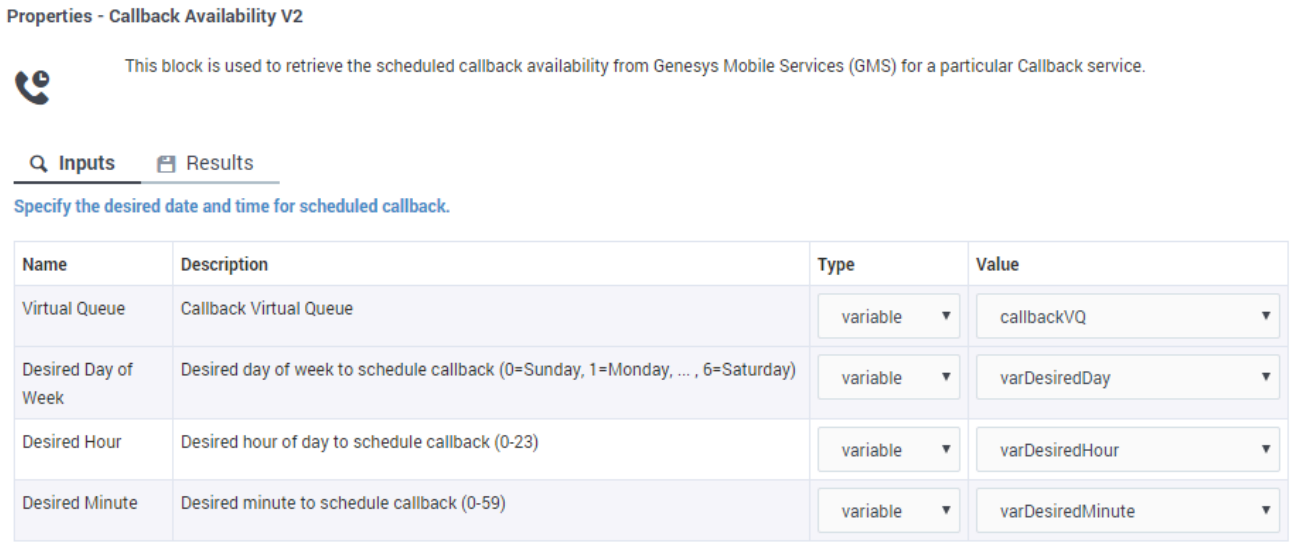
Callback Availability V2

This block checks to see if there are any available time slots for the caller's preferred callback date and time, and provides up to three possible time slots to choose from.

Inputs tab

Specify the desired date and time for the callback.

Example



Results tab

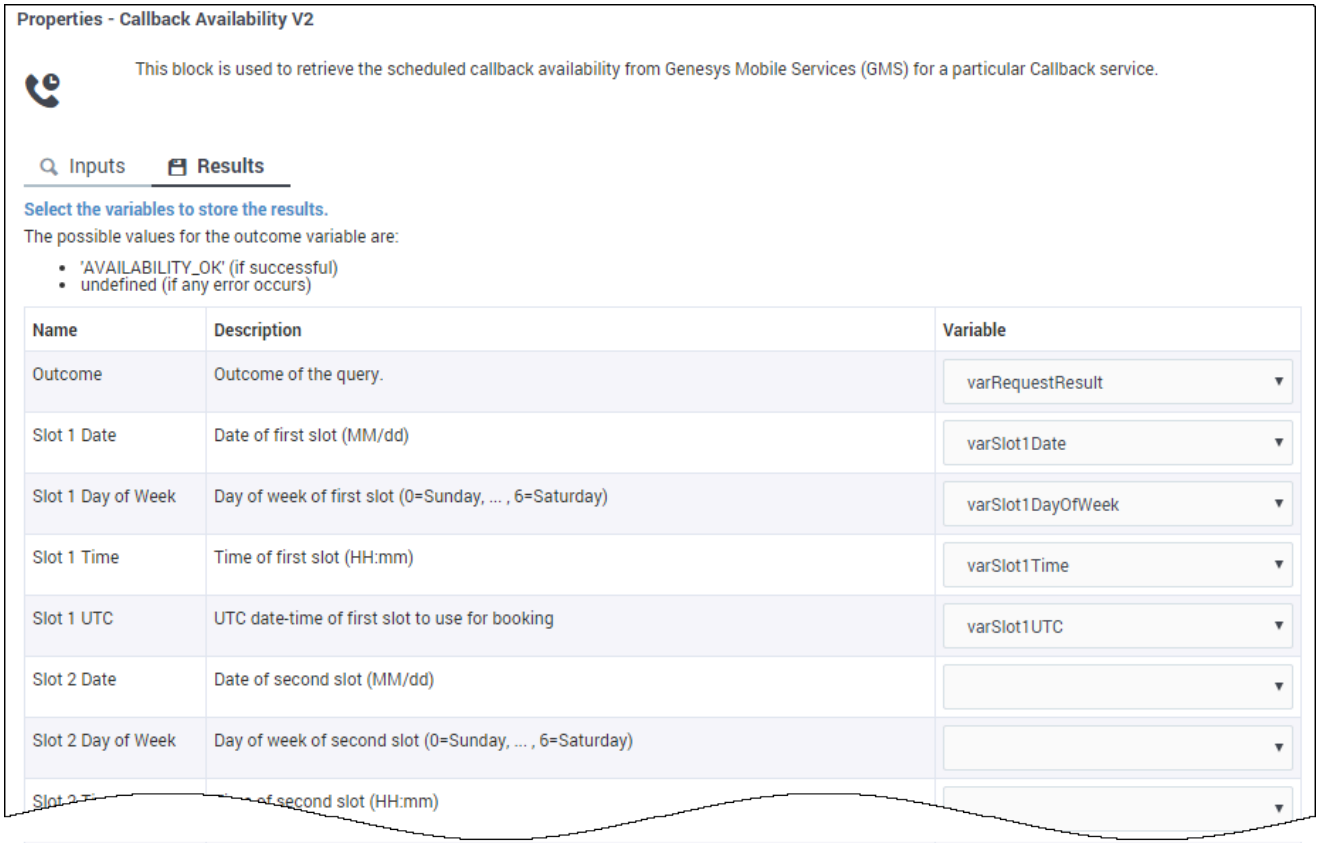
The closest available time slot to the requested callback date and time is returned, while also taking into consideration the Estimated Wait Time (for example, if the current Estimated Wait Time is 30 minutes, the earliest time slot that could be offered is 30 minutes from now). In addition, up to two alternate time slots are also proposed.

> **Important**
>
> The desired day of week, hour, and minute should be collected from the caller in the time zone of the Designer application. For the three closest time slots that are

returned, the **Slot *x* Date**, **Slot *x* Day of Week**, and **Slot *x* Time** are all in the time zone of the Designer application, so they can be played as prompts back to the customer to confirm the time, and **Slot *x* UTC** is in the UTC (Coordinated Universal Time) time zone.

You can use this tab to store the time slot results in variables.

Example



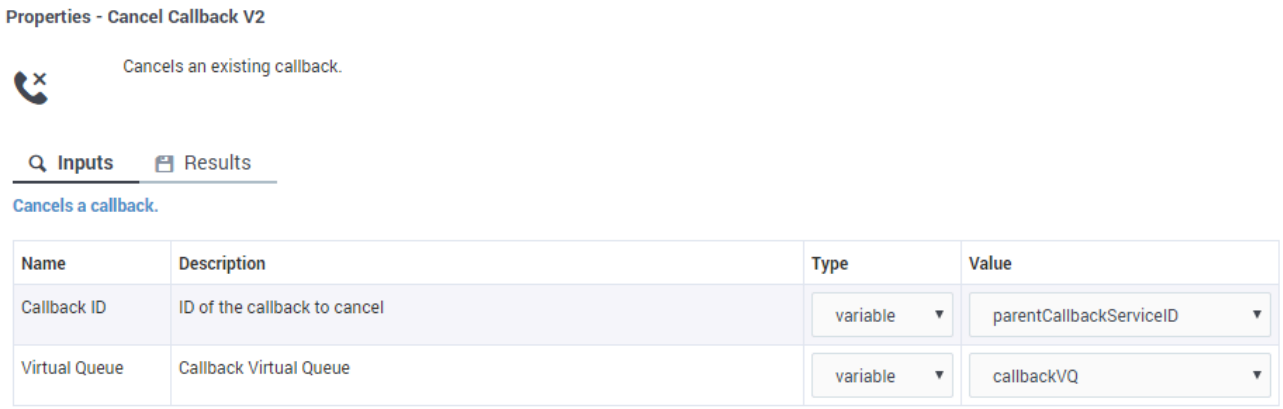
Cancel Callback V2

This block allows you to cancel a callback.

Inputs tab

Provide the **Callback ID** and **Virtual Queue** of the callback to be cancelled.

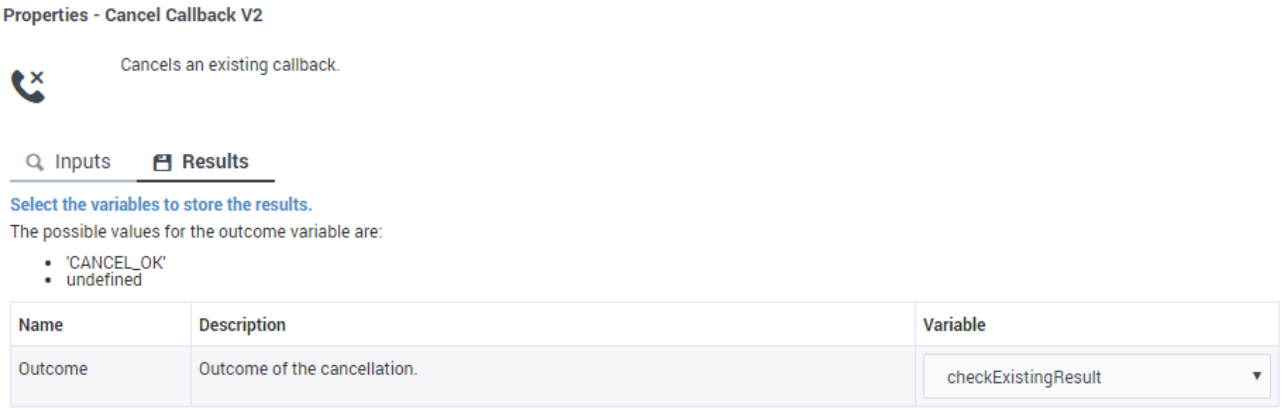
Example



Results tab

Select a variable to store the results of the cancellation request.

Example


Properties - Cancel Callback V2
Cancels an existing callback.
Inputs
Results
Select the variables to store the results.
The possible values for the outcome variable are:
'CANCEL_OK'
undefined
Name
Description
Variable
Outcome
Outcome of the cancellation.
checkExistingResult

Check for Existing Callback V2

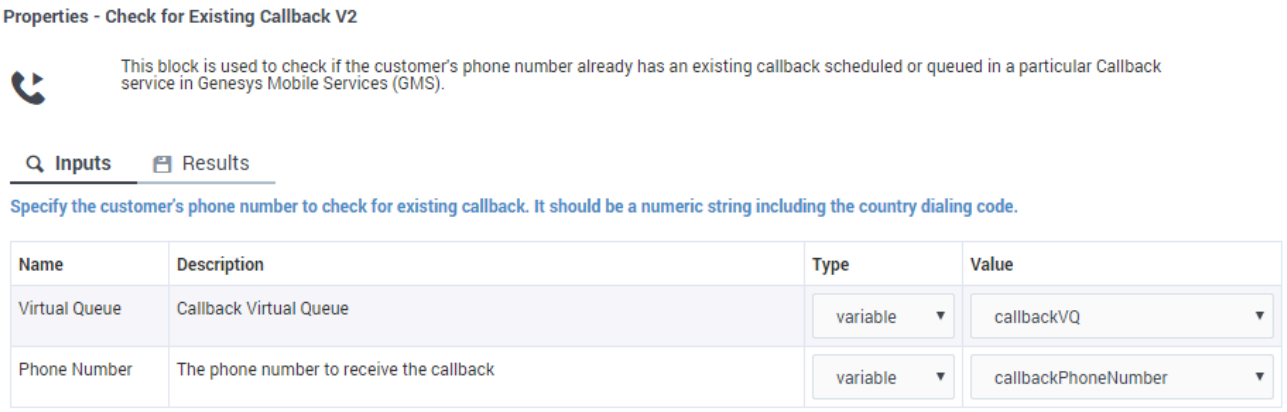
This block checks to see if an existing callback already exists for a caller's phone number in the specified virtual queue.

> **Important**
>
> This check is performed separately for each virtual queue. Keep in mind that if a caller is using different virtual queues, they could potentially book multiple callbacks with the same phone number.

Inputs tab

Provide the **Virtual Queue** and **Phone Number** that are to be checked for existing callbacks.

Example

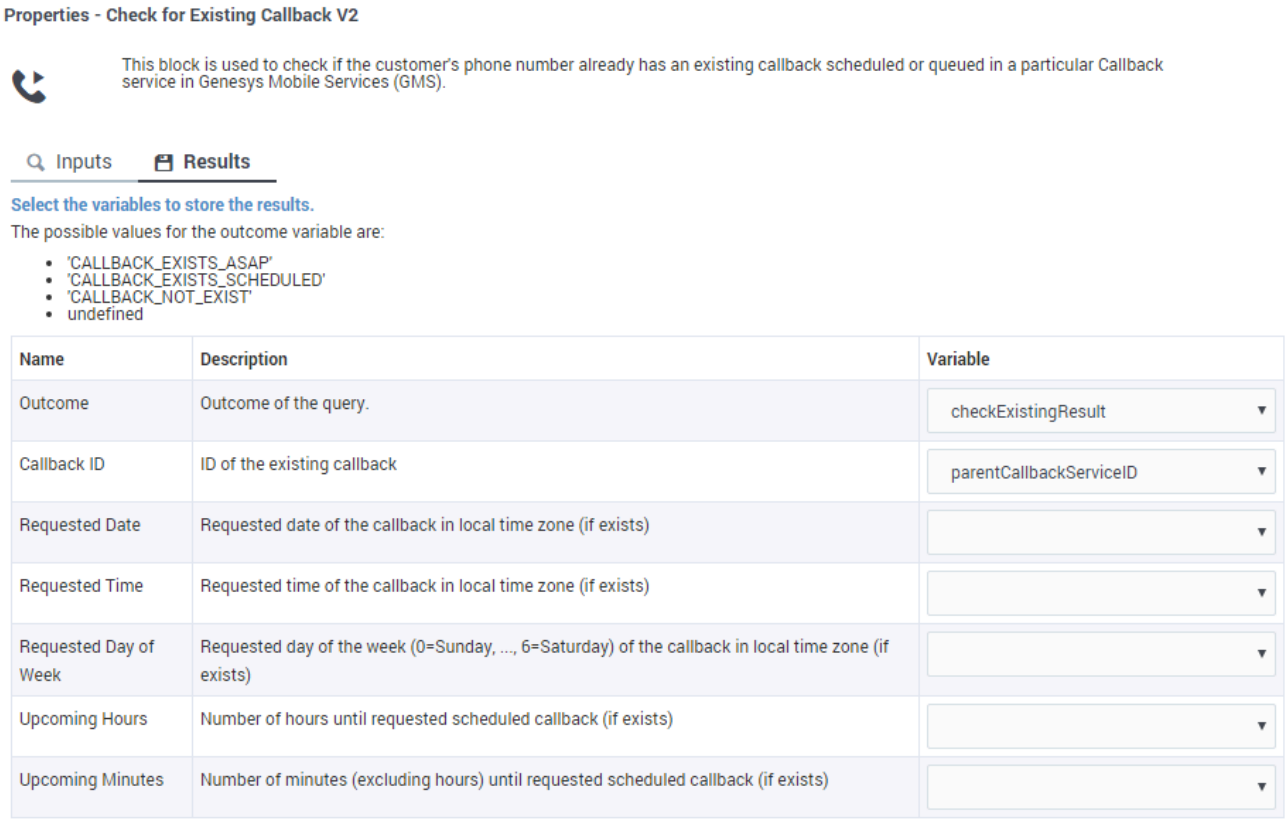


Results tab

If an existing callback with the same phone number is found in the same virtual queue, the **Callback Service ID** associated with the existing callback is returned, along with additional information such as the request date, request time, and so on.

You can use this tab to store this information in variables.

Example



Properties - Check for Existing Callback V2

This block is used to check if the customer's phone number already has an existing callback scheduled or queued in a particular Callback service in Genesys Mobile Services (GMS).

Inputs **Results**

Select the variables to store the results.

The possible values for the outcome variable are:

- 'CALLBACK_EXISTS_ASAP'
- 'CALLBACK_EXISTS_SCHEDULED'
- 'CALLBACK_NOT_EXIST'
- undefined

Name	Description	Variable
Outcome	Outcome of the query.	checkExistingResult
Callback ID	ID of the existing callback	parentCallbackServiceID
Requested Date	Requested date of the callback in local time zone (if exists)	
Requested Time	Requested time of the callback in local time zone (if exists)	
Requested Day of Week	Requested day of the week (0=Sunday, ..., 6=Saturday) of the callback in local time zone (if exists)	
Upcoming Hours	Number of hours until requested scheduled callback (if exists)	
Upcoming Minutes	Number of minutes (excluding hours) until requested scheduled callback (if exists)	

Validate Phone Number Block

This block provides phone number validation and international phone number support for Callback V2.

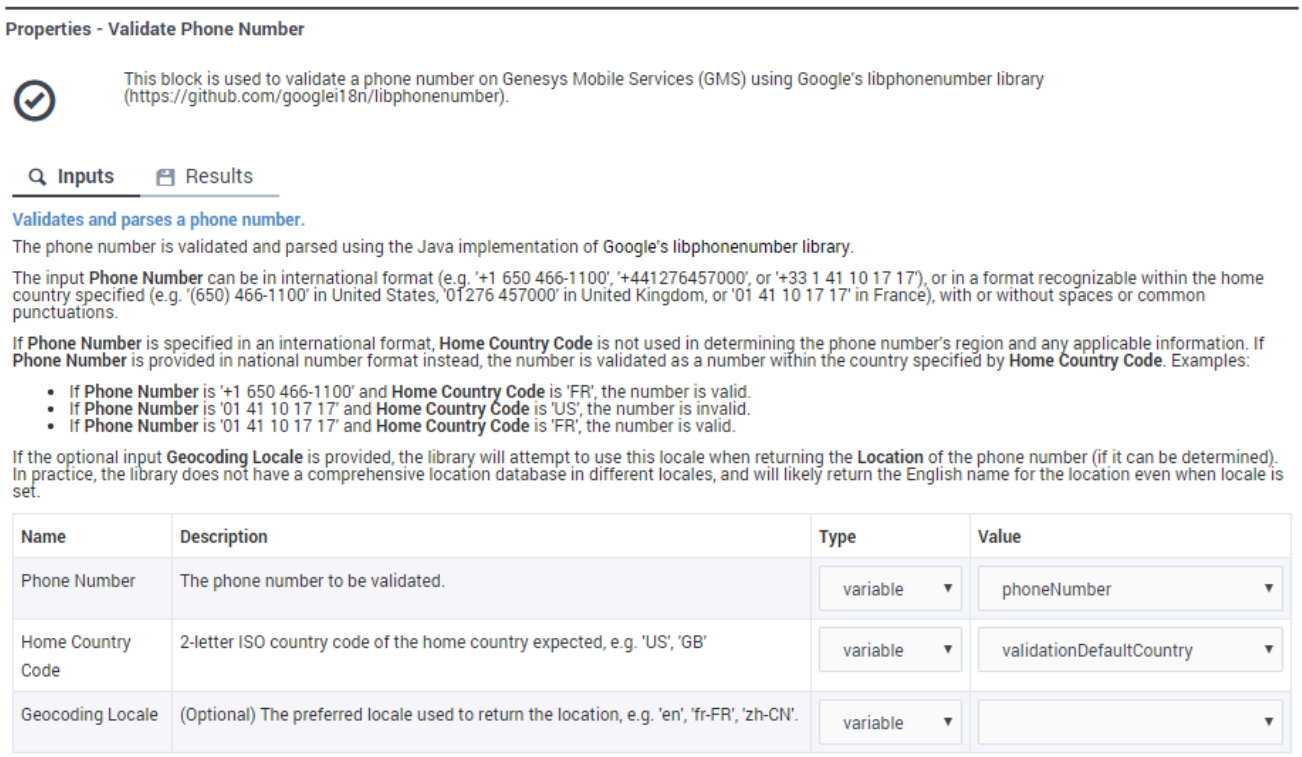
> **Important**
> This block is only supported for environments using shared GMS.

Inputs tab

The **Validate Phone Number** block has three inputs. Enter the values, or select the appropriate variables.

- **Phone number** – The phone number to be validated.
- **Home Country Code** – The 2-letter ISO code of the expected "home" country. For example, *US* or *GB*.
- **Geocoding Locale** – (Optional) The preferred locale in which to return the detected location. For example, *en* or *zh-CN*. The default is *en-US*.

Example



Results tab

Select the variables that will store the results of the phone number validation query.

All outputs are optional. If the phone number is not valid, all outputs (other than **Outcome**) will return null.

Number Validation Configurations Data Table

Callback V2 uses a special data table called NUMBER_VALIDATION_CONFIGURATIONS to provide support for phone number validation and international phone numbers.

You can view the settings for this data table by selecting it on the Data Tables page.

Parameters

This data table contains the following parameters:

Setting	Description
Config Name	Name of the configuration.
Validation Enabled	If number validation is enabled.
Default Country	The default country to be used for validation. (This should also be used for the **Home Country Code** in the Validate Phone Number block.)
Expects International Number	If callers will be prompted to include a country code when entering their phone number. **Important** If set to `true`, any countries involved in international shared-cost (such as +808) or toll-free (such as +800) numbers that are not the default country must be added to **Additional Countries Allowed**.
Always Say Country on Confirm	The country name will always be stated when confirming a phone number to the caller, even if it is the same as the home country. (It is always stated if the country name differs from the default country.)
Say Country When Not Allowed	The country name will be stated if a phone number is not from an allowed country, or use a generic message (such as "your country"). **Tip** You can set this option to `false` if you do not want your voice talent to record the names of all possible countries
Always Says International Number	When confirming a phone number to the caller, always state it in full international format. For example, say "1<xxx>5551100" instead of "<xxx>5551100"'.
Premium Rate Allowed	Whether premium rate numbers (such as 1-900 numbers in the U.S.) are allowed.
Toll Free Allowed	Whether toll-free numbers (such as 1-800 numbers in the U.S.) are allowed.
Shared Cost Allowed	Whether shared-cost numbers (such as +808 numbers) are allowed.
Voicemail Allowed	Whether voicemail numbers (if they can be determined) are allowed.
Pager Allowed	Whether pager numbers (if they can be determined) are allowed.
Additional Countries Allowed	Select the countries, in addition to the default country, which are allowed.

Callback VQ Watermark Block

This block enables you to check the number of callbacks that are queued for a specific virtual queue (VQ). The result is returned as a "watermark" value that represents the number of callbacks that are currently active (such as being dialed or routed to an agent) or waiting to be connected.

> **Important**
> This value does not include scheduled or completed callbacks.

You can add this block to the top level of a default type application.

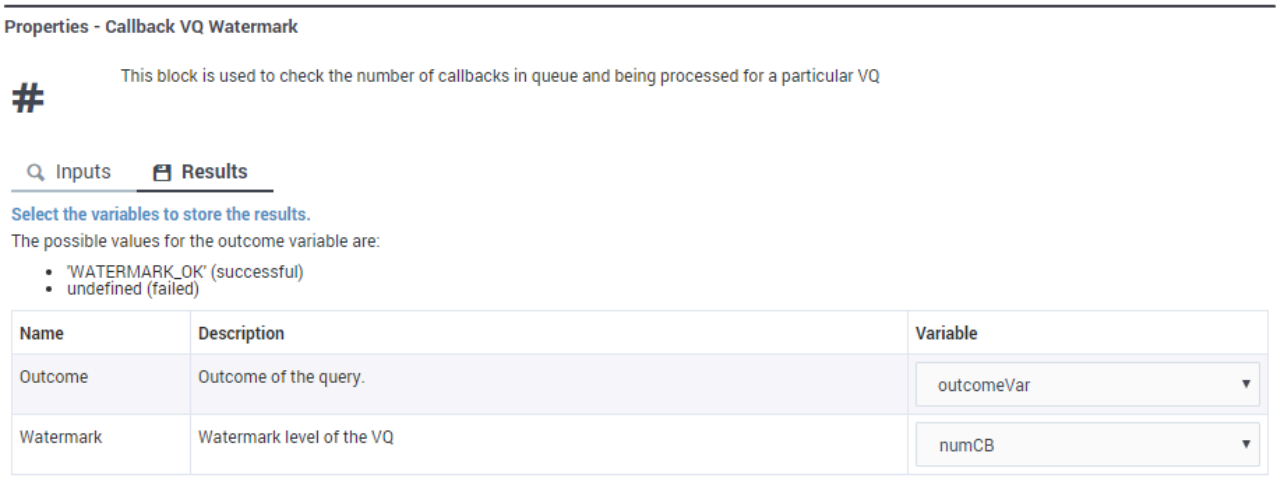
For example, let's say you wanted to stop offering immediate callbacks when a certain number of calls are already in the queue. You could add this block to do a watermark value check for the specified VQ, and then use a Segmentation block to decide what to do when a certain value is reached.

Inputs tab

Specify the Callback VQ to check for the number of callbacks that are being dialed, routed to an agent, or waiting to be connected.

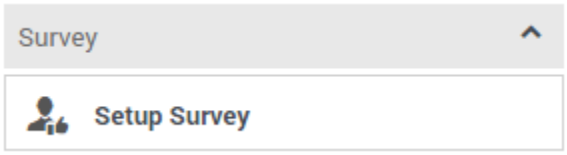
Results tab

Specify the variables in which to store the results of the query.


Properties - Callback VQ Watermark
This block is used to check the number of callbacks in queue and being processed for a particular VQ
Inputs
Results
Select the variables to store the results.
The possible values for the outcome variable are:
'WATERMARK_OK' (successful)
undefined (failed)
Name
Description
Variable
Outcome
Outcome of the query.
outcomeVar
Watermark
Watermark level of the VQ
numCB

Survey Blocks

These blocks are used to manage surveys offered to callers.



Use the links below to learn more about each block.

Setup Survey

Sets up a survey for the caller.

Used in: **Initialization**, **Self Service**, **Assisted Service** *

* Surveys are typically offered during the **Self Service** phase and completed by an accepting caller after they have finished speaking with an agent in the **Assisted Service** phase. But if you have set up your survey to be **Immediate**, the caller can complete the actual survey while still in the **Self Service** phase of the application.

Setup Survey Block

You can use the **Setup Survey** block in the **Assisted Service** phase to set up a survey for the caller.

Typically, you offer the survey earlier in the call, in either the **Self Service** phase or before routing begins in the **Assisted Service** phase. Then, once the caller has been served, place the **Setup Survey** block in the **Assisted Service** phase to provide the survey functions.

Once the block is set, you can choose to start the survey immediately (the caller completes the survey within the **Self Service** phase of the current application), or after the caller has finished talking to an agent (if they agreed earlier to take the survey, the caller is then sent to a number assigned to a survey application).

You can also choose to not start the survey if the caller rejects the offer or to not offer the survey at all.

Using this Block

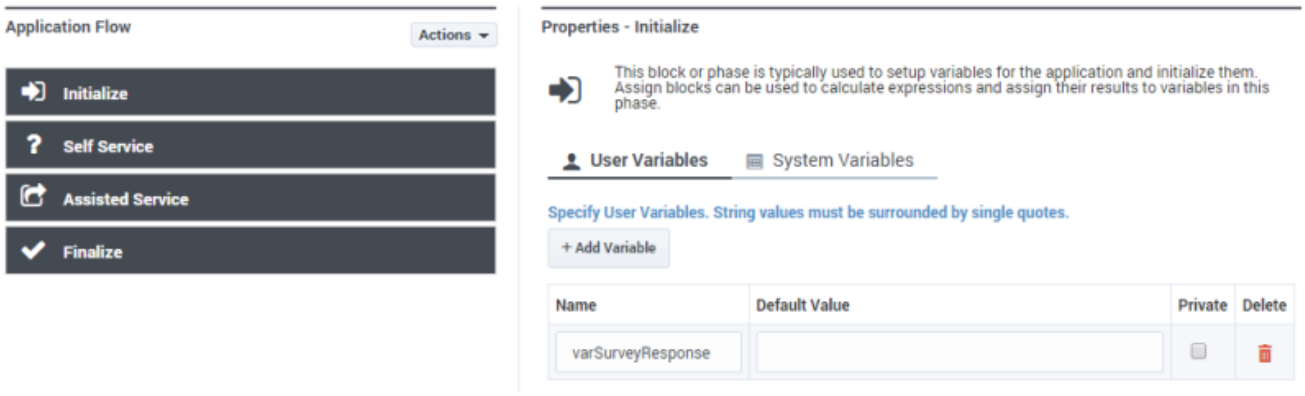
In most applications, you will place a User Input block before the **Setup Survey** block and use prompts to ask the caller if he wants to take a survey. You can then use a Segmentation block to segment the call based on the caller's response.

The sections below explain how to incorporate a survey into an existing application. Your application and User Variable names might differ.

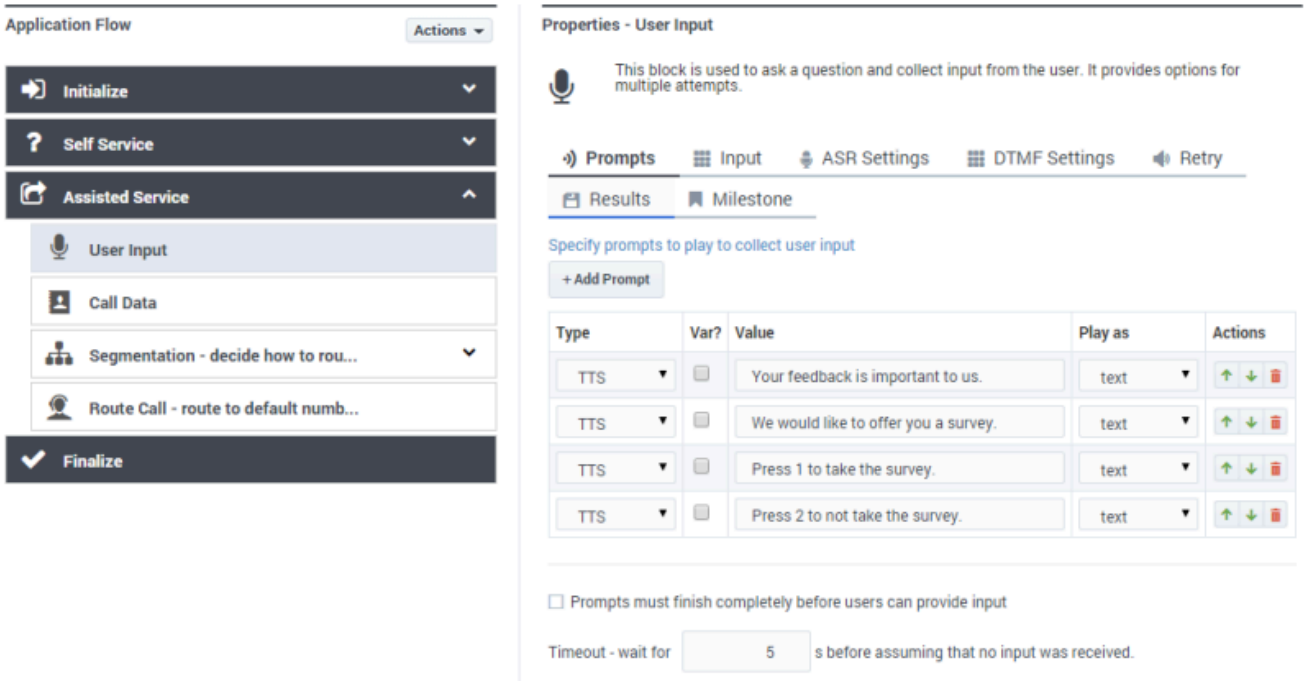
> **Important**
>
> The examples below offer the survey in the **Assisted Service** phase, but it is also possible to offer the survey in the **Self Service** phase. In either case, the **Setup Survey** block must be placed in the **Assisted Service** phase.

Offer the Survey

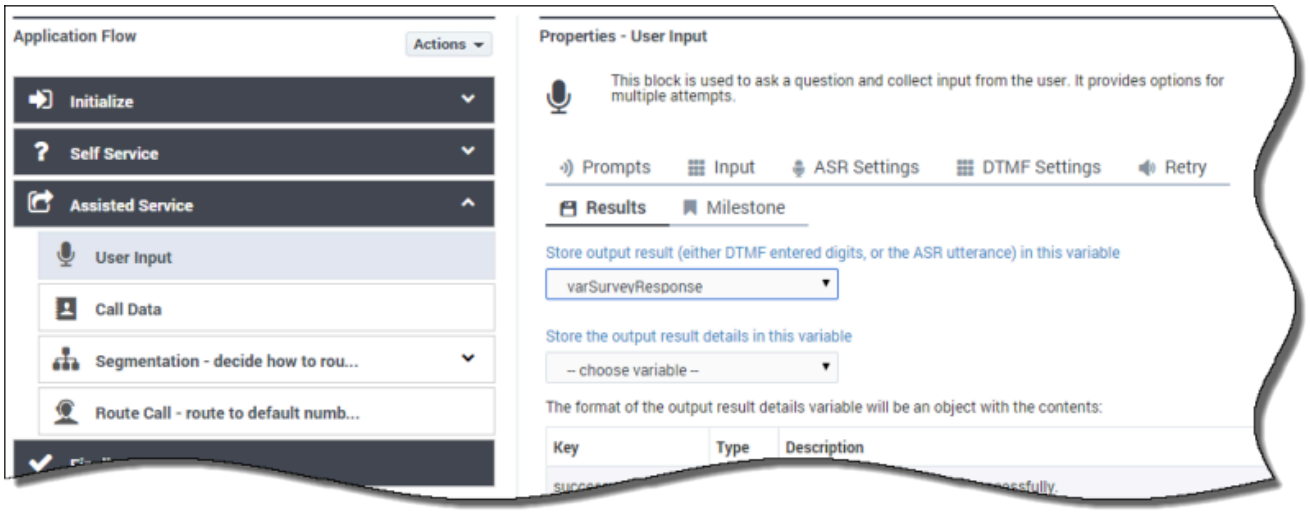
Click the **Initialize** phase and create a **User Variable** named **varSurveyResponse**.



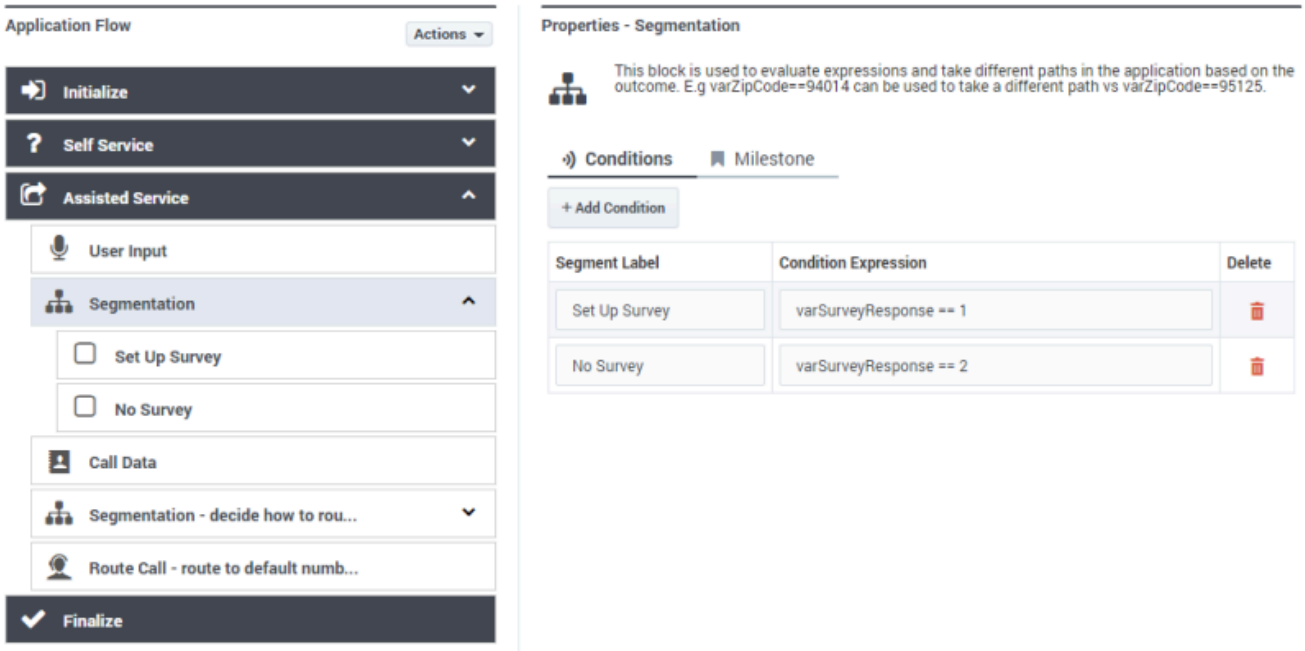
In the **Assisted Service** phase of your application, before the call is routed, add a **User Input** block and create a message in the **Prompts** tab. In this example, you can use the following values:



Next, in the **Results** tab, select the **varSurveyResponse** variable that you created earlier. This variable stores the input from the caller.

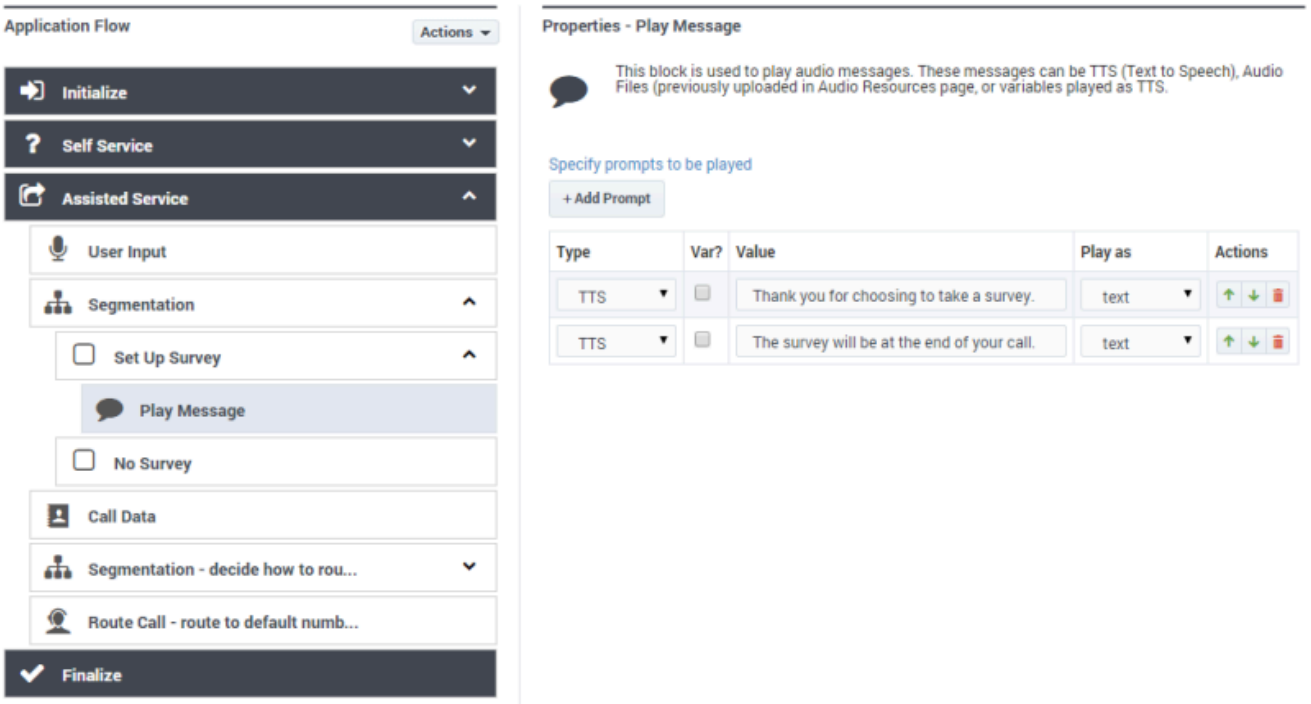


Next, place a Segmentation block to configure how your application responds to the result from the **User Input** block. In this example, configure the **Segmentation** block as shown below:

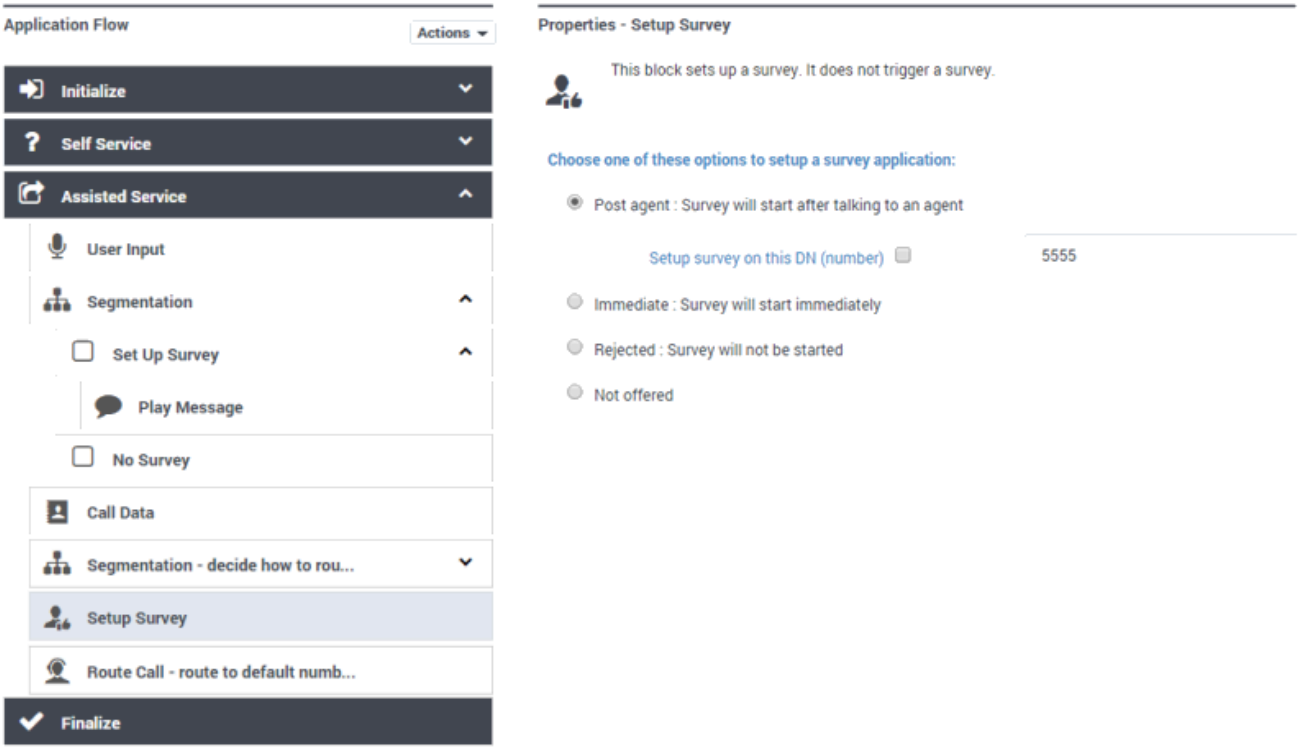


Set Up Survey Segment

The application processes the **Set Up Survey** segment if the caller pressed **1** to accept the survey. Next, the application uses a Play Message block to thank the caller for taking the survey.



Next, place a **Setup Survey** block before the call is routed to an agent. Select the **Post agent: Survey will start after talking to an agent** option and enter the DN of the survey application. The example shown below uses the DN 5555, but your survey application might use a different DN. Optionally, you can enable the check box to specify the DN as a variable.

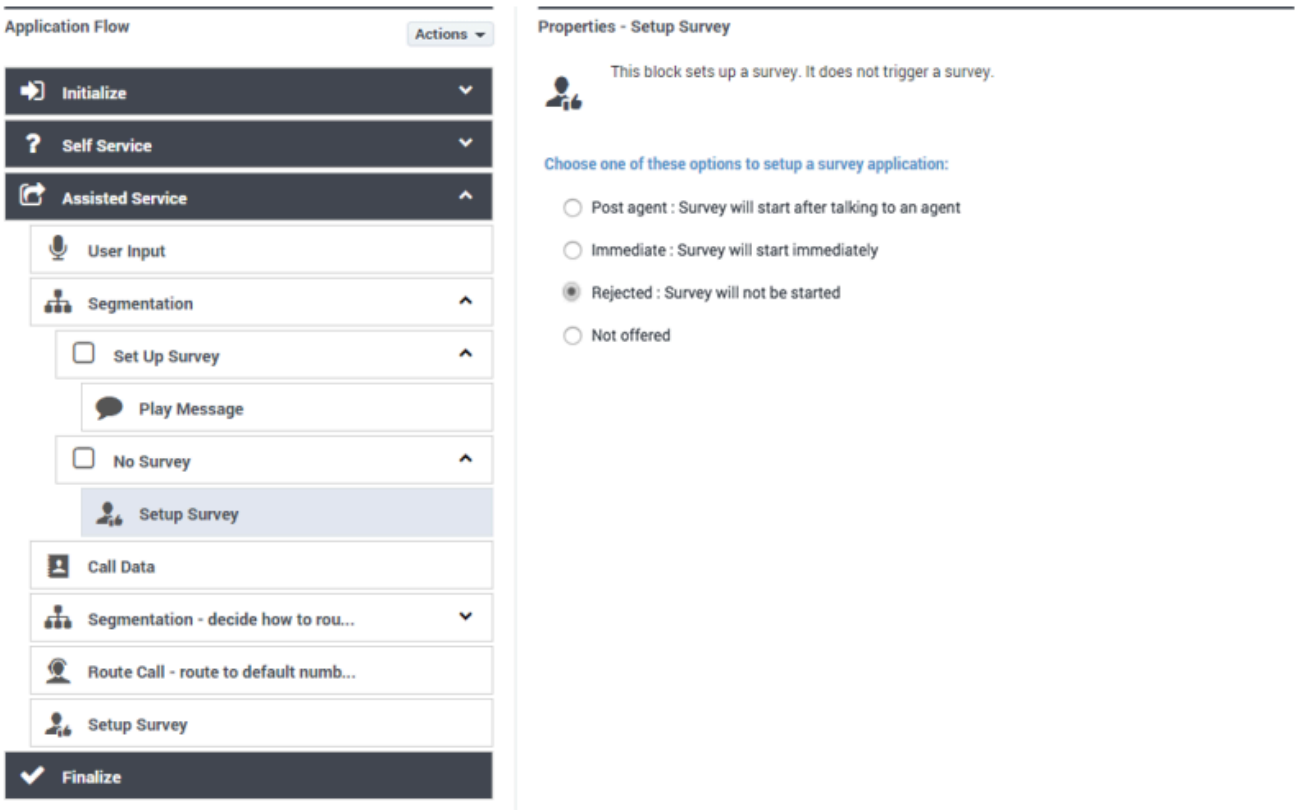


The call forwards to the survey application. See the Creating the Survey Application section for more information.

No Survey Segment

The application processes the **No Survey** segment if the caller pressed **2** to decline the survey.

Place a **Setup Survey** block and select **Setup was offered but it was rejected**.



Survey Not Offered

You might have noticed that a third option exists in the **Setup Survey** block - **Setup was not offered - no need to setup survey**.

For reporting, this option records that the caller was never offered a survey. This can happen for several reasons. For example, the caller might have ended the call early or in the **Self Service** phase, or your application might contain a segment in which it does not make sense to offer a survey.

To receive reporting in these scenarios, you must place a **Setup Survey** block in your application and select the **Setup was not offered - no need to setup survey** option to record that this interaction did not include a survey offer.

Creating the Survey Application

The actual survey takes place in a second application. This application is loaded on the number that you specified in the **Setup Survey** block.

A survey application is created with the application type **Default** and behaves in the same way as other applications. You can use **User Input** blocks to ask questions and record responses. Each **User Input** block stores the response from the caller for reporting.

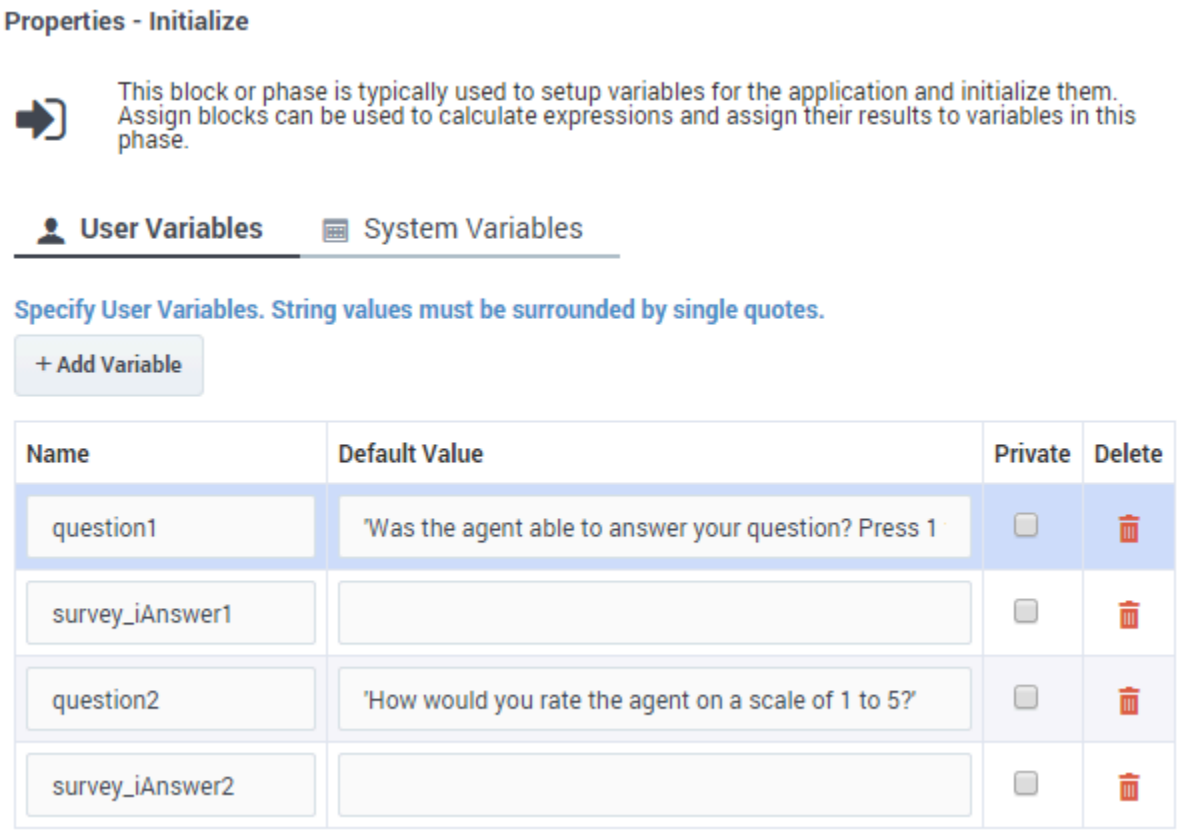
> **Tip**
>
> As survey applications are **Default** type applications, you can use Route Call and various other blocks to direct the call if the customer's survey responses meet certain criteria. For example, if the caller inputs a low satisfaction score, you can use a Segmentation block to check for low satisfaction scores and a **Route Call** block to route the call to an agent to follow up on the customer's concerns.

Example

The following example demonstrates how to create a simple survey application.

First, create a new application of type **Default** to provide the survey.

In the application, create a series of variables to hold the questions and answers for your survey. In the example below, **question1** and **question2** hold the question that the survey asks the caller, and **survey_iAnswer1** and **survey_iAnswer2** holds the answer from the caller.



Designer also provides standard variables, which you can view in the **System Variables** tab, that you can use if your company uses standard reporting. For example, instead of using **survey_iAnswer2** to hold the feedback score for the agent, we could instead use **survey_iAgentScore**.

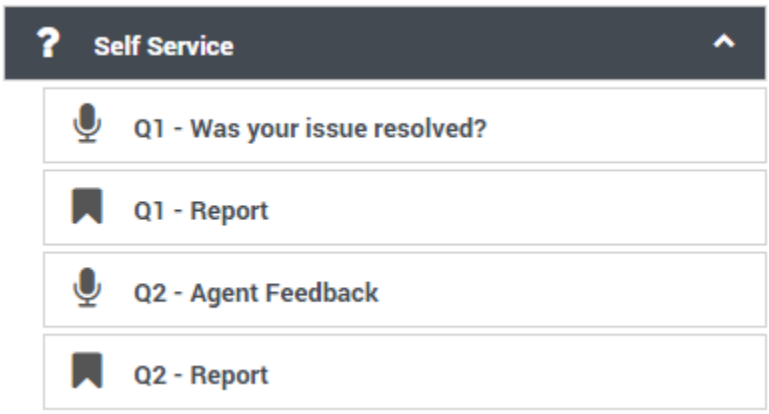
Variable	Editable	Purpose
survey_sOffer	No	Specifies whether a survey was offered, accepted, or rejected. This variable is set by the **Setup Survey** block.
survey_iRecommendScore	Yes	A rating (on a scale from 0 to 10) that indicates if the company, product, or service is recommended. This variable is used for calculating the Net Promoter Score (NPS).
survey_iAgentScore	Yes	Specifies a user satisfaction score for the agent (if this question is asked in the survey).
survey_iCompanyScore	Yes	Specifes a user satisfaction score for the company (if this question is asked in the survey).
survey_iCallScore	Yes	Specifies a user satisfaction score for the entire call (if this question is asked in the survey).
survey_iProductScore	Yes	Specifies a user satisfaction score for the product (if this question is asked in the survey).
survey_sQ1..10	Yes	You can create these variables (1-10) to store **string**-type survey responses that will be used for reporting. (Use the naming convention as shown. For example, *survey_sQ1*, *survey_sQ2*, and so on.)
survey_iQ1..10	Yes	You can create these variables (1-10) to store **integer**-type survey responses that will be used for reporting. (Use the naming convention as shown. For example, *survey_iQ1*, *survey_iQ2*, and so on.)

Important

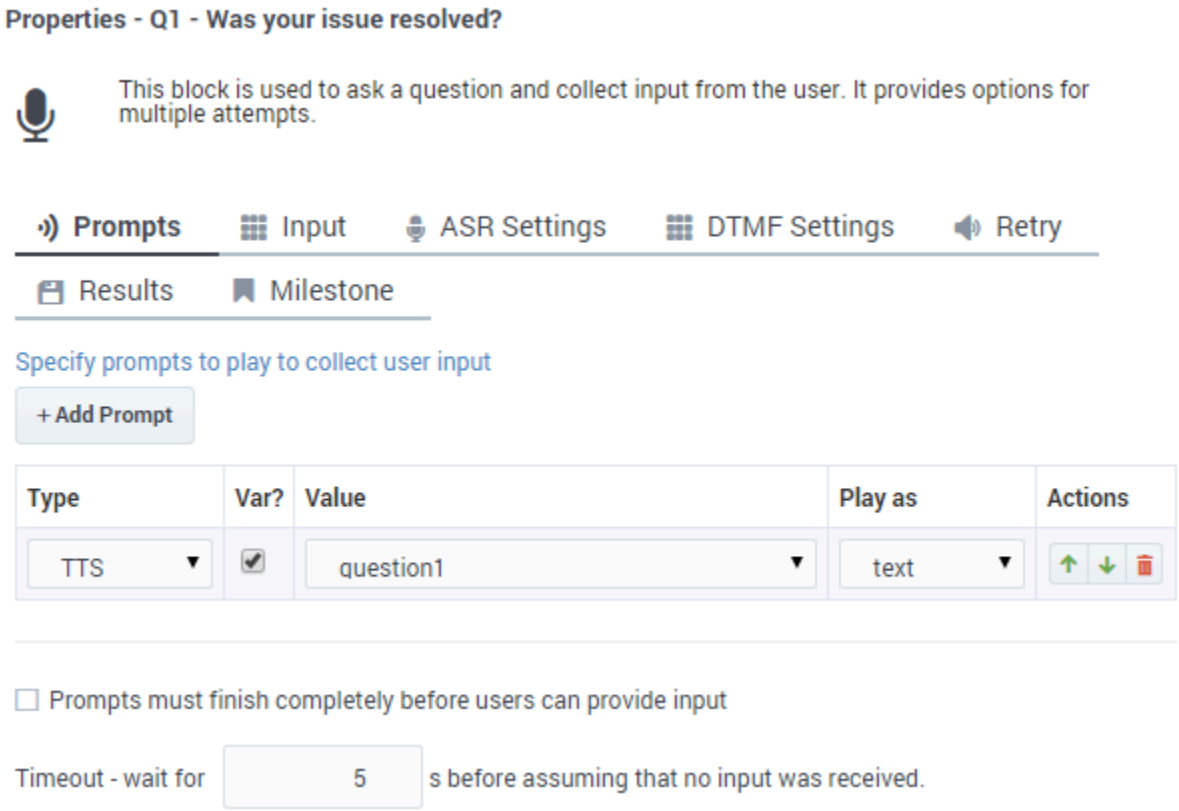
Survey answer variables must use the following naming convention:

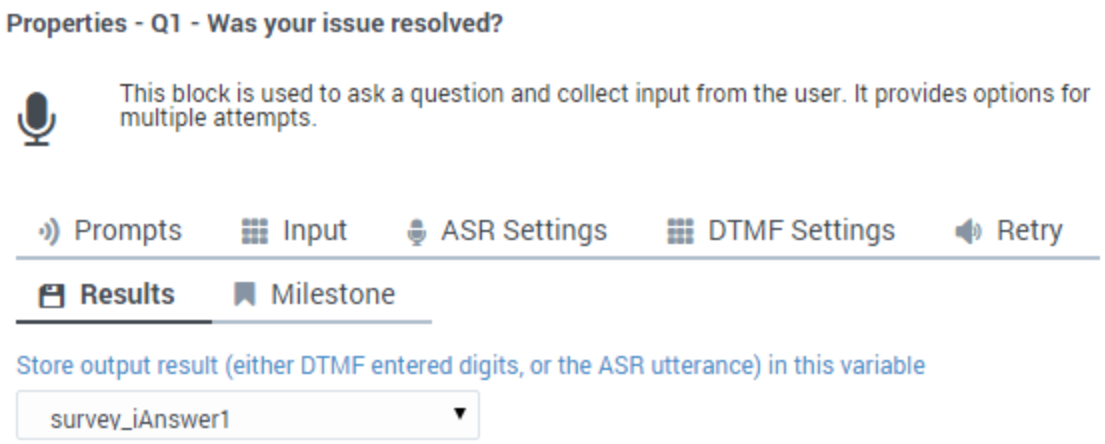
- The name must have the prefix `survey_`.
- The next character must indicate the data type (for example, `i` for integer or `s` for string).
- Example: `survey_iAnswer`.

Next, we add a series of User Input blocks and Milestone blocks to the **Self Service** phase. The **User Input** block asks the survey question and the **Milestone** block reports the survey answer.

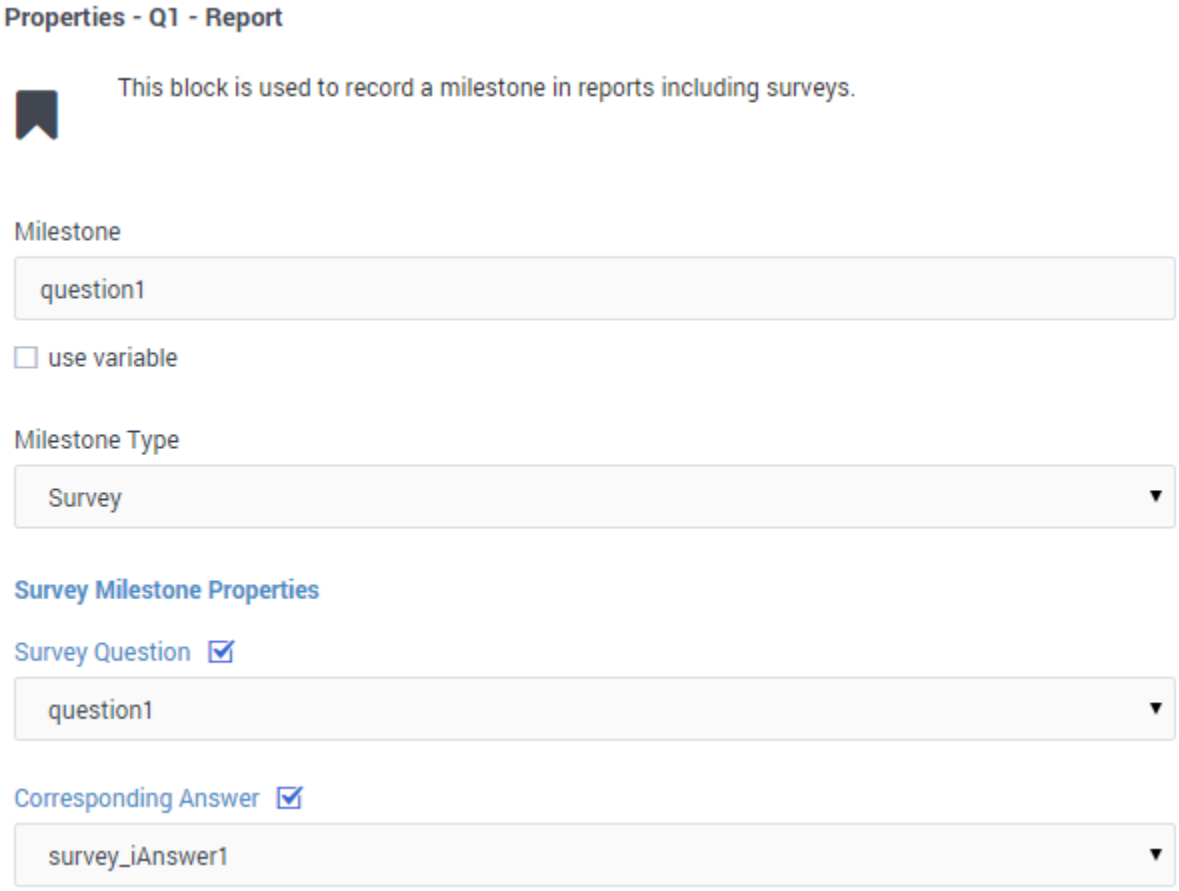


In each **User Input** block, select the question variable in the **Prompts** tab and answer variable in the **Results** tab.

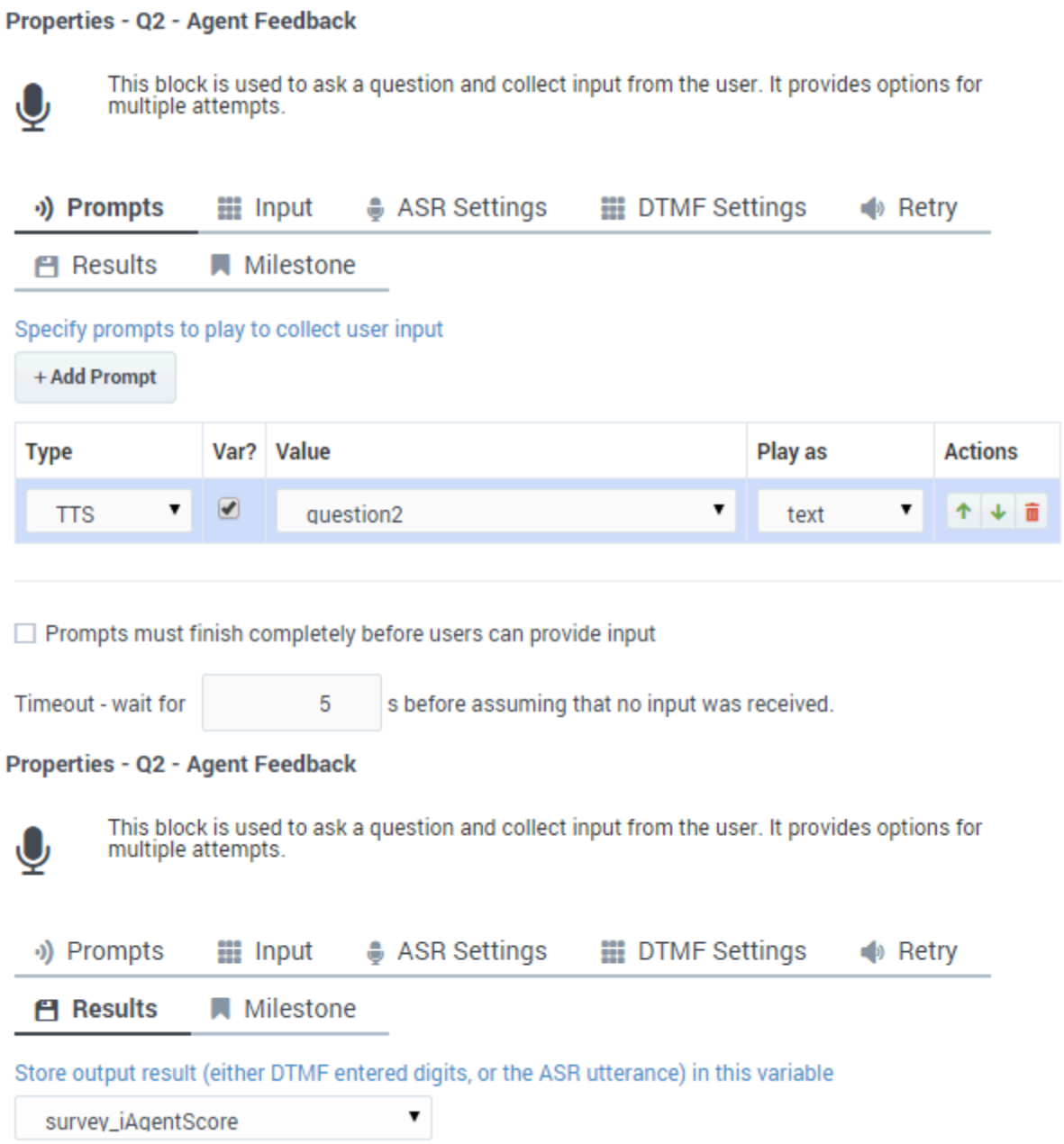




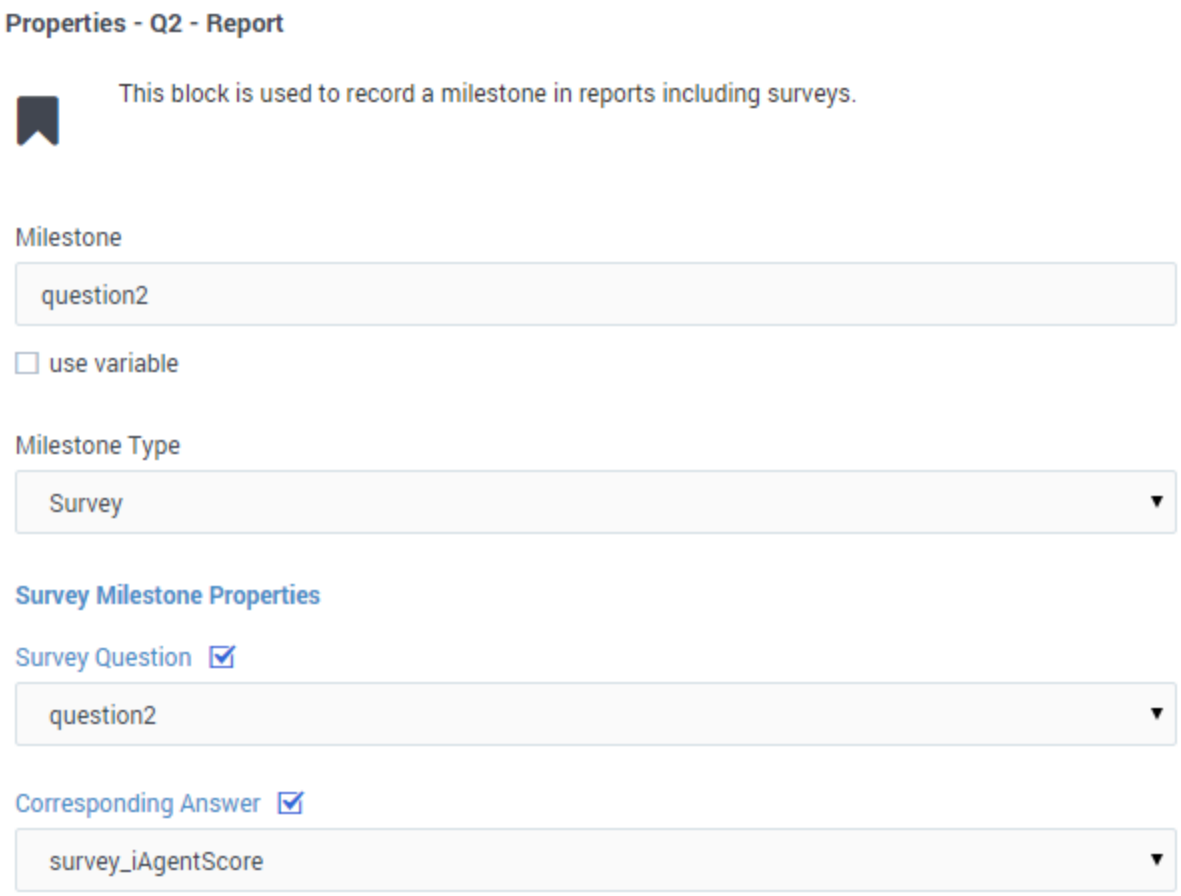
In each **Milestone** block, select the question and answer to send to reporting.



The following graphics show the process for survey question two, using the standard answer variable **survey_iAgentScore**.


Properties - Q2 - Agent Feedback
This block is used to ask a question and collect input from the user. It provides options for multiple attempts.
Prompts
Input
ASR Settings
DTMF Settings
Retry
Results
Milestone
Specify prompts to play to collect user input
+ Add Prompt
Type
Var?
Value
Play as
Actions
TTS
question2
text
Prompts must finish completely before users can provide input
Timeout - wait for
5
s before assuming that no input was received.

Properties - Q2 - Agent Feedback
This block is used to ask a question and collect input from the user. It provides options for multiple attempts.
Prompts
Input
ASR Settings
DTMF Settings
Retry
Results
Milestone
Store output result (either DTMF entered digits, or the ASR utterance) in this variable
survey_iAgentScore


Properties - Q2 - Report
This block is used to record a milestone in reports including surveys.
Milestone
question2
use variable
Milestone Type
Survey
Survey Milestone Properties
Survey Question
question2
Corresponding Answer
survey_iAgentScore

Analytics

What is it?

Designer **Analytics** is a powerful tool that gives you deeper insight into how your applications are being managed on the platform. It features a series of dashboards, each of which offers rich visualizations and in-depth reporting panels that highlight specific aspects of your operations.

Key features:

- Almost real-time reporting means that as soon as an application session ends, Designer Analytics can start using the data to build reports.
- 90-day data retention, so you can see how your applications are performing over time.
- Customization options, so you can change how the panels are displayed.
- Advanced filtering options, so you can focus on the data you want to see.

What can it do?

Designer Analytics provides a rich overview of your contact center operations. It lets you track calling trends, monitor how callers are interacting with the applications, and quickly notice and react to any potential issues with the applications or system platform.

It answers questions like:

- How long are callers waiting to speak with an agent?
- How many calls did we get yesterday? Last week? Last month?
- How many callers are taking our survey?
- Are callers navigating ok? Are they getting stuck anywhere?
- What percentage of our calls are from North America? Europe? Asia?

The **Analytics** dashboards not only provide useful information about your callers, they can also help you maximize productivity and make your call flows more efficient.

Go to the Dashboards page to learn more about the different types of dashboards.

Dashboards

Designer includes several system dashboards that you can start using right away.

Each dashboard contains reporting panels that focus on a particular aspect of your operations. For example, panels might display results based on milestones, system errors, or the paths that callers took through an application.

Many of the panels also have options for viewing additional details about the data displayed (such as the query used to generate the results) or for changing the panel properties.

Session Detail Records

The data contained in the Session Detail Records (SDR) is the "secret sauce" that Designer uses to generate the reporting panels shown on the dashboards.

Each time an interaction is processed by an application, Designer creates a SDR. The fields within the SDR capture important details about the interaction, such as the starting time of the call, source and destination numbers, the block sequence (or path) that the caller took through the application, and the final status of the call (for example, the caller hung up or was connected to an agent).

> ### Important
> **Sessions vs. Calls**: A *session* is not the same thing as a *call*. Sessions are started each time a call (or interaction) is processed by an application. If an interaction is processed by multiple applications (or processed multiple times by the *same* application), multiple SDRs are created.

Designer assigns each interaction a unique ID that follows it through each session that is created, thus enabling you to track the entire journey of an interaction from start to finish, across each application that handles it. This makes SDRs useful for call flow analysis and troubleshooting.

Dashboard types

The following system dashboards are included with Designer and ready to use. Use the dashboard icons to quickly navigate between the different types:

Summary lets you see at a high-level how your application sessions are being handled across the platform.

Application Details gives you a closer look at how callers are moving through the application flows, such as milestones reached, activities completed, and paths taken.

Durations shows you how much time callers are spending in various parts of the applications.

Data Tables displays disposition information for your applications in a spreadsheet-like table format.

Spikes shows the peaks in your application session counts over a specific period of time.

Heatmap uses rectangles of various sizes and colors to show the intensity in occurrence (or "heat") of a particular item or event.

Path lets you visualize how callers are moving through an application.

Sankey Path Analysis is similar to Path, but generates the results based on milestones and menu inputs.

Sunburst Path Analysis renders your reporting data as a sunburst graphic.

Inputs shows you how callers are responding to the various menu options.

Surveys gives you a deeper look into the performance of your survey applications.

External Services provides details about external requests made by the applications.

Routing Analysis lets you analyze routing sessions.

Business Controls provides details about Business Controls objects.

Session Detail Records lets you view some of the raw data contained in the Session Detail Records (SDRs).

Summary

The **Summary** dashboard gives you a quick overview of what's happening with your application sessions.

You can use the information provided here to see if there are any patterns that can give you insight into the sessions being initiated on Designer. For example, you might look at sessions over a period of time (such as the last 24 hours, a week, or a month) to see how sessions are being handled across various applications, what's happening to those sessions, and how much time callers are spending in various stages of applications.

Reports on this dashboard

Count Over Time

This report shows you the patterns of traffic that are coming onto the platform over a period of time. You can quickly adjust the range of time shown, say, for the last hour, 24 hours, or the last week, to see how sessions are being initiated on the platform.

Each bar indicates the total number of sessions for each application that took place during the given time period. The higher the bar, the more sessions that ran. You can easily see what time of the day (or what day of the week) that traffic is higher or lower, and organize your business accordingly.

Total

The total number of sessions or interactions that were processed.

System Errors

The percentage of sessions (out of all sessions) that had system errors. (A system error does not necessarily mean that your callers experienced any issues with the applications. This report helps you to notice any possible issues with the system platform.)

Abandoned

The percentage of sessions that are being abandoned while callers are in the Assisted Service (or routing) phase. A higher number might indicate that callers are waiting too long and hanging up before they can be connected to an agent.

Duration > 10 Mins

The percentage of sessions where a caller spent more than 10 minutes combined in both the Self-Service and Assisted Service phases. This gives you an overall look at how long it's taking for callers to be processed.

Routing > 5/10 Mins

The percentage of sessions where a caller spent more than 5/10 minutes in the routing (or Assisted Service) phase. This lets you see if there are any potential issues that might be causing callers to be stuck in the routing phase for too long.

Count by Apps

This report provides distributions of sessions across the various applications that were built on Designer. This data could be useful in allocating resources based on the traffic across applications.

Count by Disposition

This report provides distributions of sessions across the various disposition codes. A *disposition* represents the status of a call at the time it exited the call flow, such as whether it was routed to an agent or the caller hung up.

This data could be useful in finding out how your application sessions are actually performing. For example, if a high amount of sessions are getting abandoned in Self Service, you can check if there is an application error or some other reason why those calls aren't getting sent to agents.

[+] More about dispositions

default

System Error
There was an unexpected error in the application (such as a script validation error).

Application Timeout
The application got stuck in a loop and reached the timeout limit.

Terminated - <reason>
The call was ended due to a certain condition, such as:

- **Terminate Call** — the application reached a Terminate Call block.
- **Business Hours** — the call came in outside of regular business hours (if set up this way in the Business Hours block).
- **Special Days** — the call came in on a special day (if set up this way in the Special Day block).
- **Emergency** — the emergency flag was set (if set up this way in the Emergency block).
- **Menu Option** — the caller chose a menu option to exit or end the call.

Abandoned in Self Service
The caller hung up before completing the Self Service phase of the application.

Abandoned in Queue
The caller completed the Self Service phase, but hung up while waiting to speak with an agent.

Completed in Self Service
The caller completed their call in the Self-Service phase.

Routed to Agent
The call was successfully delivered to an agent.

Routed to DN
The call was successfully delivered to a direct number.

Count by DID

This report provides details about the distributions of sessions across various Dialed Numbers (DNIS). Typically, a DNIS represents a department or line of business.

Dispositions by App

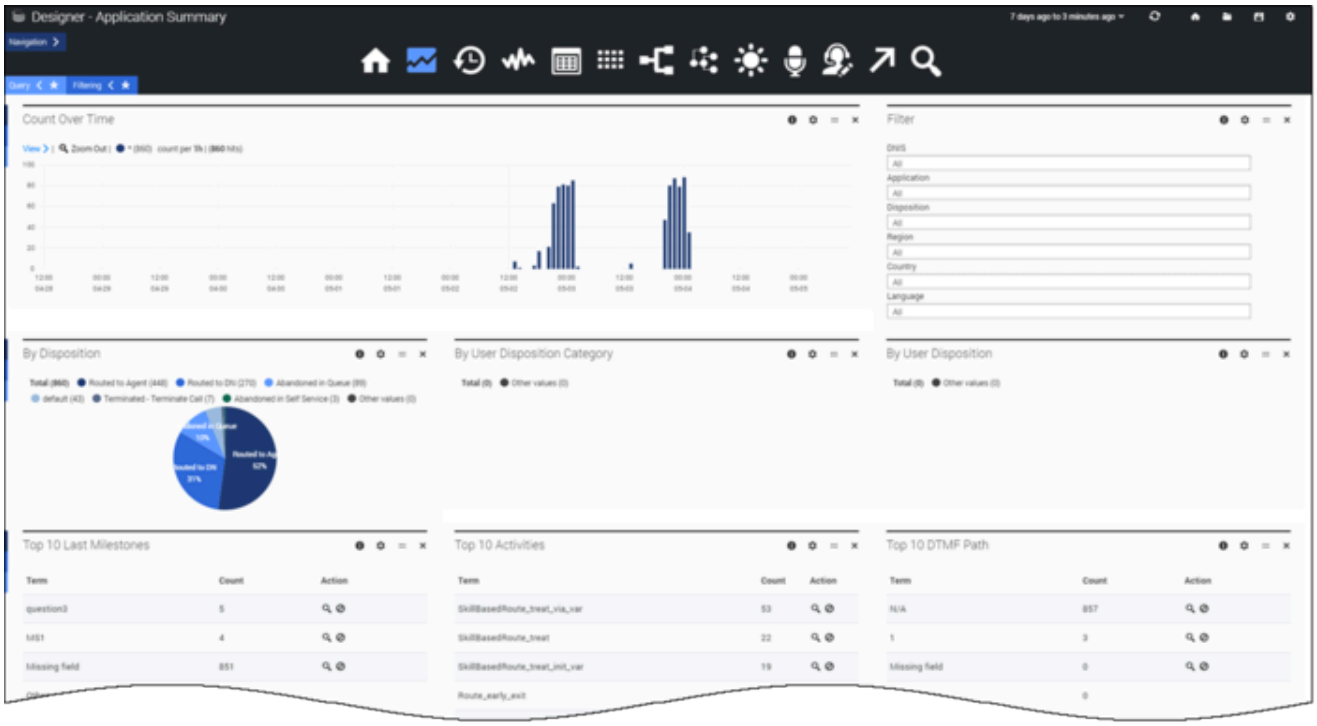
This report provides distributions of sessions across two parameters, **Disposition Code** and **Applications**. This give you a quick look into how sessions are distributed across the various

applications and what their disposition codes are.

This could be very handy if you want to compare sessions with certain disposition codes across several applications. For example, if **Abandoned in Queue** for Application A is higher than that of Application B, you might want to think about adjusting the resources assigned to handle calls coming into Application B.

Application Details

The **Application Details** dashboard gives you a closer look at how callers are moving through the application flows.



Most of the reports on this dashboard are focused on the milestones reached, activities completed, and paths taken by callers as they move through the application flows. You can quickly see if there are any unexpected deviations (such as a sudden rise or drop in certain milestones) that might require further investigation.

Reports on this dashboard

Count Over Time

(See the Summary dashboard for a description of this report.)

Filter

The **Filter** panel appears on many of the dashboards (notable exceptions include the **Session Detail Records** and **Summary** dashboards). Use it to select the specific values you want to filter for, such as **Application** or **Disposition**.

> **Important**
> Any filters you select are applied across *all* of the dashboards, not just the one you are viewing.

You can also toggle the Filtering tab to see any filters that are currently applied.

By Disposition

This report shows the distributions of **final dispositions** across all calls received. A *final disposition* is the status assigned to a call at the time it exited the call flow, such as whether it was routed to an agent, terminated due to it being a special day or outside of regular business hours, or the caller hung up. (Learn more about dispositions here.)

By User Disposition Category

This report shows the distributions of final dispositions across all calls received, by the disposition category. The *disposition category* is the high-level status of the call when the caller exited the call flow, such as **Transfer** or **Abandoned**.

By User Disposition

This report shows the distributions of final dispositions across all calls received, by user disposition. The *user disposition* is the status assigned to a call when the caller exited the call flow. (Learn more about dispositions here.)

Top 10 DTMF Path

Designer automatically tracks various DTMF paths (how callers are responding to things like menu options), so if the top paths here are not the ones you expect, it might indicate that callers are not following the intended flow.

Top 10 Last Milestones

This report tracks the last milestones that callers reached before the call ended. A *milestone* is a custom benchmark (or checkpoint) that you've defined in an application to indicate that a significant point in the application flow was reached. For example, you might set up a milestone to mark when callers have made a successful payment, and another for when they've agreed to certain terms and conditions.

There are also other milestone-related reports:

- **Top 10 Bailout Milestones** — when the caller requests an agent (for example, by pressing **0**)
- **Top 10 Self-Helped Milestones** — when the caller is able to complete their call in self-service
- **Top 10 Deflection Milestones** — when the caller is not able to complete their call in self-service and is sent elsewhere (for example, to an agent)

Top 10 Activities

An *activity* is a task that you've defined in an application as a series of steps with a starting point and stopping point. For example, you might set up an activity for making a payment that starts with the caller being asked for their credit card details and then ends with the system sending those details to a payment processor and receiving the approval.

Each activity has a *start* and *end* point, and can be complete or incomplete, with success or failure.

Top 10 Milestones

This report displays the top ten milestones that were successfully reached by callers.

Top 10 Deflection Messages

These messages are generated when a caller can't complete their call in self-service and is redirected elsewhere.

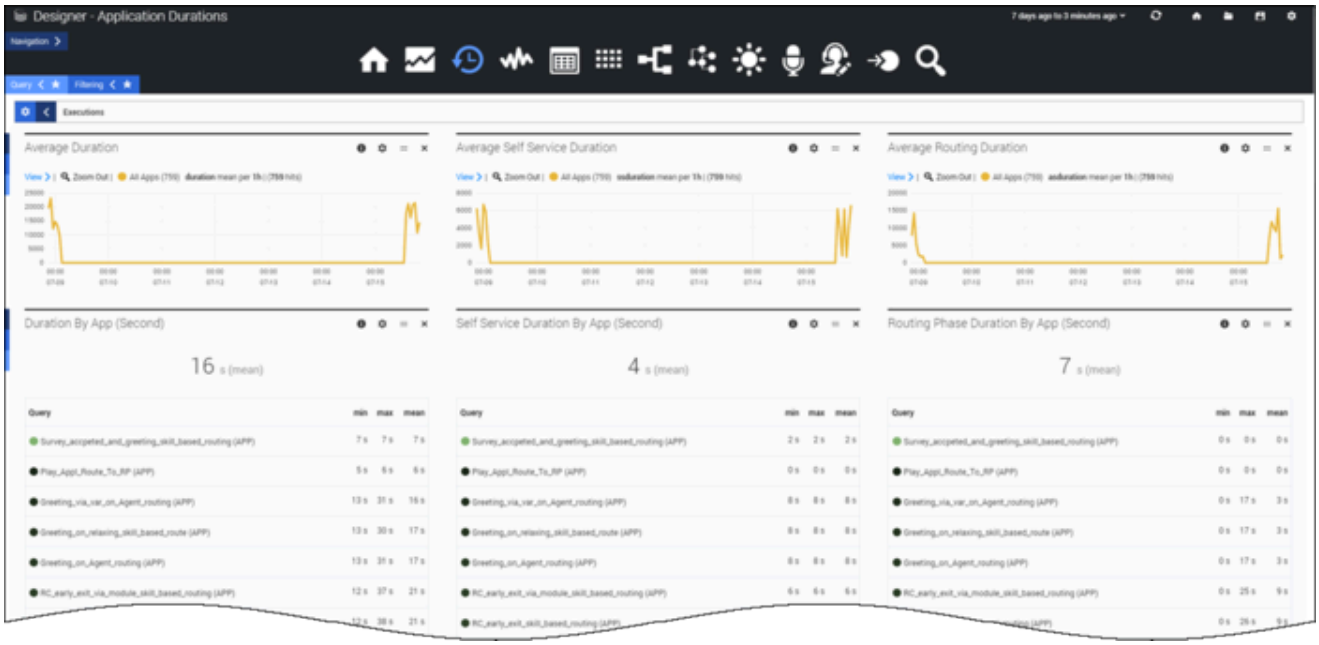
Performance by Activities

This report shows a breakdown of performance data for each activity over a given time period, grouped by Type (**user** or **system**). Note that activities with the same name are grouped together, even if they belong to different applications.

Clicking an activity will show (or hide) the call results for that activity. You can also export the results to a CSV file.

Durations

The **Durations** dashboard shows you how much time (in seconds) your callers are spending in the different phases of the application flow.



This dashboard can quickly show you:

- if callers are spending more time than expected in the **Self Service** or **Assisted Service** phases of an application
- if the amount of time callers are spending in certain sections of an application varies at different times
- any other unusual peaks in time being spent

The reports on this dashboard can help you determine if there are any possible issues with the application or its flow design, such as callers finding it difficult to navigate.

Reports on this dashboard

Count Over Time

(See the Summary dashboard for a description of this report.)

Filter

(See the Summary dashboard for a description of this panel.)

Average Duration

The average duration of time (in seconds) for all interactions across all applications. This data is captured for every 5-minute interval and then plotted as a line graph for the last 7 days.

Average Self Service Duration

The average time (in seconds) that callers are spending in the **Self Service** phase of the application, across all applications.

Average Routing Duration

The average time (in seconds) that callers are spending in the **Assisted Service** phase of the application, across all applications.

Routing Phase Duration by App

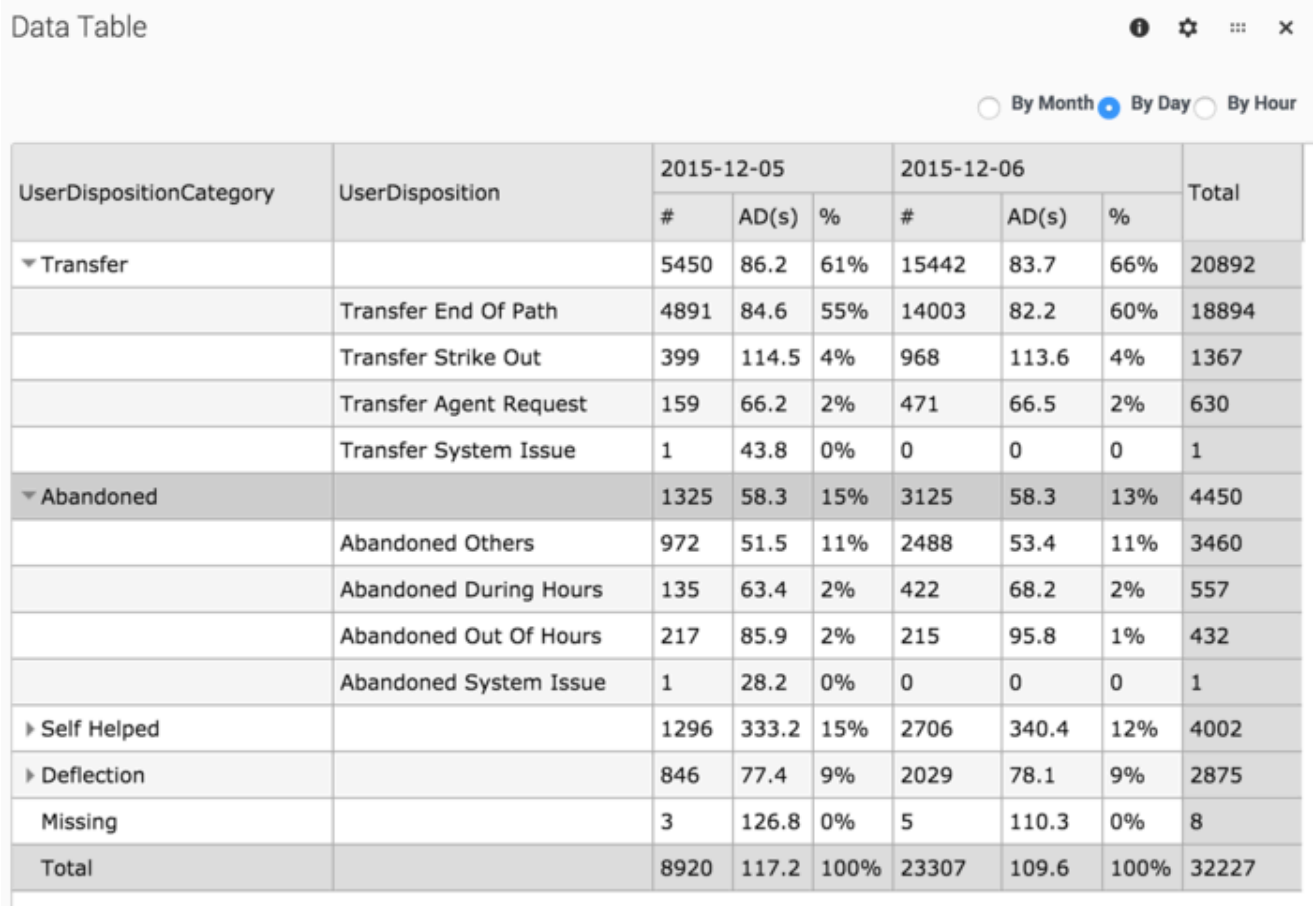
The amount of time (in seconds) callers are spending in the **Assisted Service** phases of the application, broken down by application.

Self Service Duration by App

The amount of time (in seconds) callers are spending in the **Self Service** phases of the application, broken down by application.

Data Tables

The **Data Tables** dashboard arranges the disposition data of your applications in a tabular format.


Data Table

By Month By Day By Hour

UserDispositionCategory	UserDisposition	2015-12-05			2015-12-06			Total
		#	AD(s)	%	#	AD(s)	%	
Transfer		5450	86.2	61%	15442	83.7	66%	20892
	Transfer End Of Path	4891	84.6	55%	14003	82.2	60%	18894
	Transfer Strike Out	399	114.5	4%	968	113.6	4%	1367
	Transfer Agent Request	159	66.2	2%	471	66.5	2%	630
	Transfer System Issue	1	43.8	0%	0	0	0	1
Abandoned		1325	58.3	15%	3125	58.3	13%	4450
	Abandoned Others	972	51.5	11%	2488	53.4	11%	3460
	Abandoned During Hours	135	63.4	2%	422	68.2	2%	557
	Abandoned Out Of Hours	217	85.9	2%	215	95.8	1%	432
	Abandoned System Issue	1	28.2	0%	0	0	0	1
Self Helped		1296	333.2	15%	2706	340.4	12%	4002
Deflection		846	77.4	9%	2029	78.1	9%	2875
Missing		3	126.8	0%	5	110.3	0%	8
Total		8920	117.2	100%	23307	109.6	100%	32227

Reports on this dashboard

Count Over Time

(See the Summary dashboard page for a description of this report.)

Filter

(See the Application Details dashboard page for a description of this panel.)

Data Table

This report organizes the application disposition information into a table view. It groups the dispositions by category so you can see:

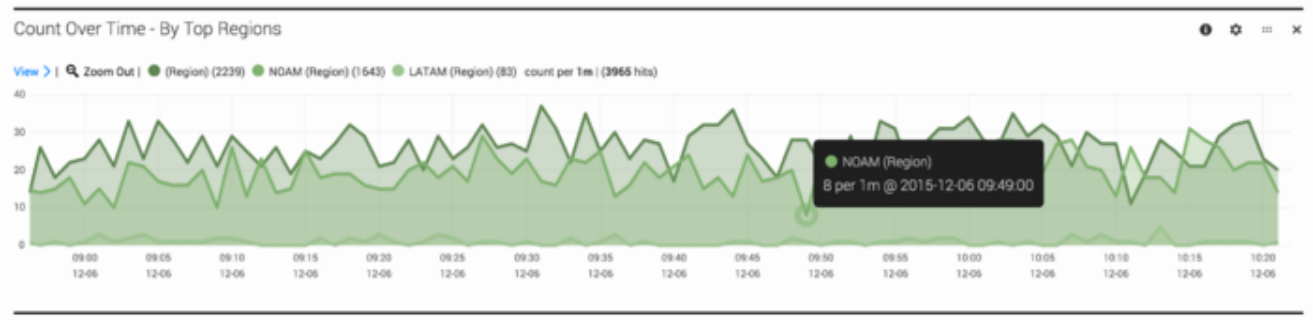
- the number of sessions that took place (**#**)
- the average duration (in seconds) for each session (**AD**)
- the percentage count of sessions (**%**)

You can then use the options to toggle the results **By Month**, **By Day**, or **By Hour** to get a more detailed look at the final results of your calls.

> ### Important
> **By Hour** is only available if the given time window is within two days.

Spikes

This dashboard provides a "spikes" view of application sessions over a given period of time. It lets you easily visualize your call volumes by breaking down the total number of sessions by **Top Regions**, **Top Countries**, and **Top Languages**.



Reports on this dashboard

Count Over Time

(See Count Over Time.)

Filter

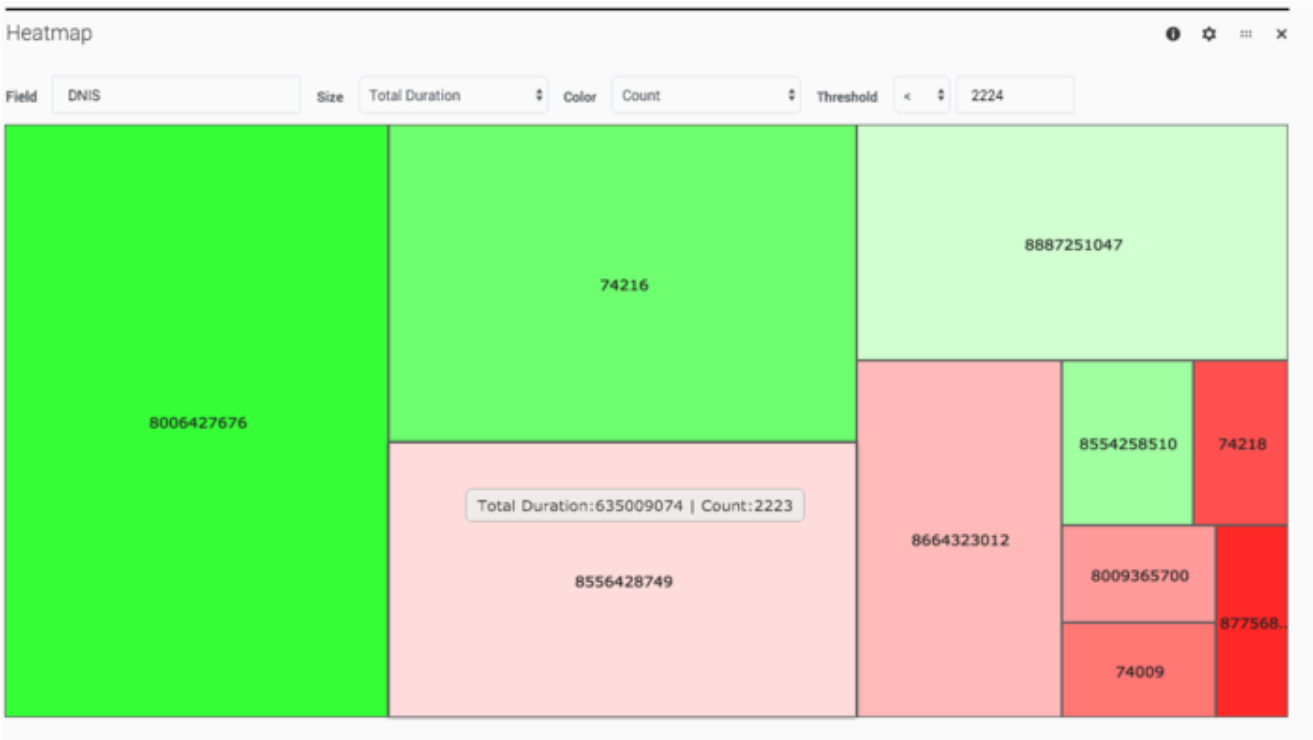
(See Filter.)

Count Over Time - By Top Regions/Countries/Languages

These panels take the total number of sessions received during the given time period and break them down into individual reports for each item.

Heatmap

The Heatmap dashboard uses colors to show a graphical representation of individual values. The graph changes colors when certain counts (or thresholds) for the selected field are reached.



Reports on this dashboard

Count Over Time

(See Count Over Time.)

Filter

(See Filter.)

Heatmap

Basically, the heatmap is a collection of colored rectangles that represent the session counts for a particular field.

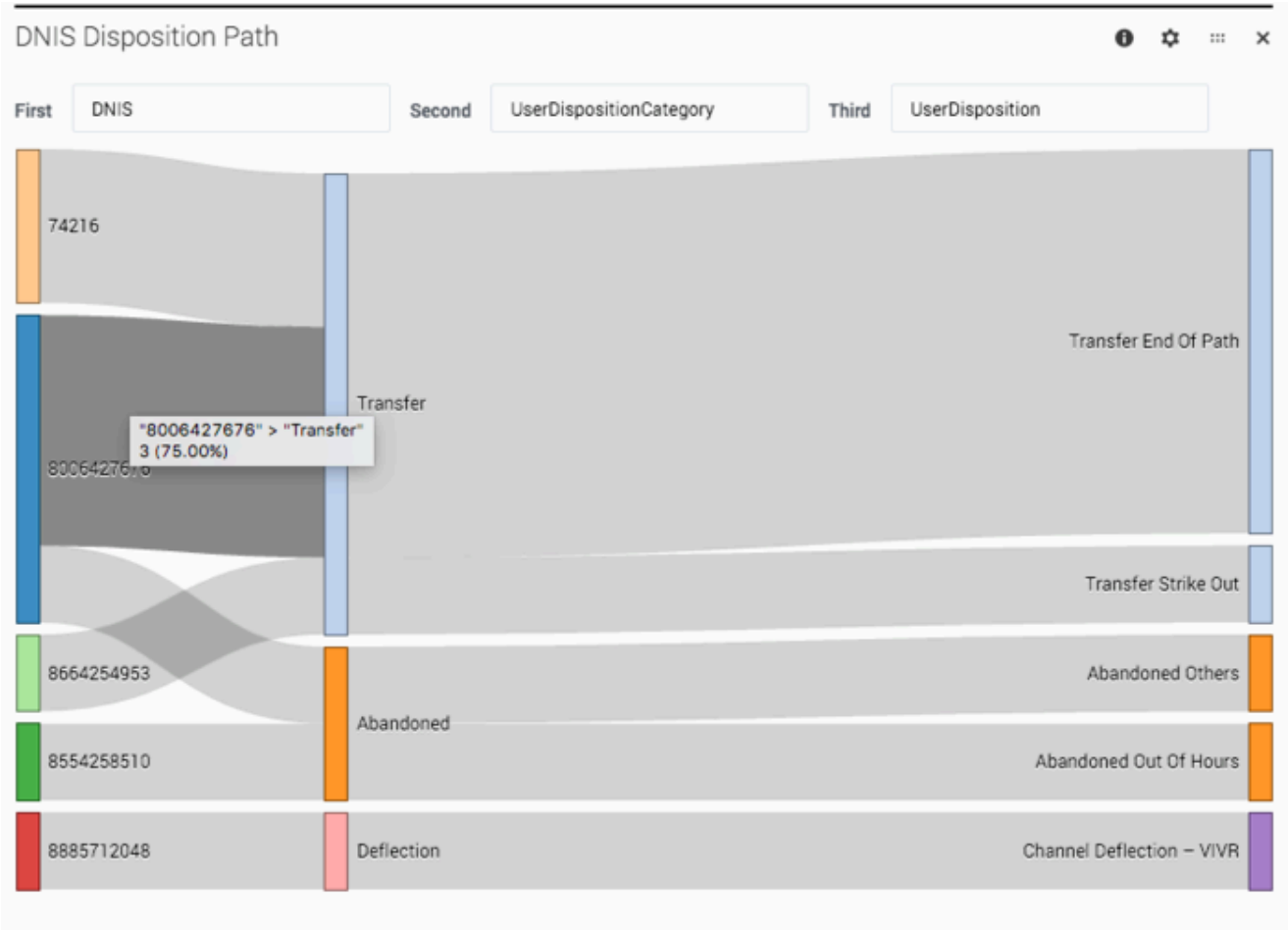
If you look at the example above, the Heatmap panel shows the session counts for the **DNIS** field, with the size and colors of the rectangles changing as the **Total Durations** values increase or decrease. The longer the **Total Durations**, the larger the rectangle, and the color progression goes from green to red based on the **Count**, with the color intensity set by the **Threshold**.

So, in this example, the largest rectangle (DNIS=8006427676) had the longest **Total Duration**, while the smallest (DNIS=87568) had the shortest. As the **Threshold** is set to **< 2224**, rectangles where the count is less than that are in red. The deeper the red, the further away the count is from the set threshold.

Path

This dashboard provides a visualization of how callers are moving through the applications by tracing their path through various nodes.

> **Tip**
> The Sankey panel might not be able to render correctly if there are several nodes or links to display. When this happens, you'll be prompted to increase the height of the panel.



Looking at this sample report, there are three nodes selected:

- First = *DNIS*, which is the number that callers dialed

- Second = *UserDispositionCategory*, which is the top-level disposition category (for example, **Transfer**)
- Third = *UserDisposition*, which is how the call finally ended up within that disposition category (for example, **Transfer End of Path**)

For each application session that took place during the selected time, a line is drawn between each node to represent the path that callers took. As more sessions share a common path, the path gets thicker.

You can easily see how callers are navigating through the application flows and quickly adjust the selected nodes to build paths for other values and categories.

Reports on this dashboard

> **Tip**
> You can select **Final Disposition** or **End Call Path** as an option to track callers in the application.

Count Over Time

(See Count Over Time.)

Filter

(See Filter.)

DNIS Disposition Path

This path shows how the sessions are tracking by dialed number through the high-level disposition categories (such as **Transfer** or **Abandoned**) to a final disposition (such as **Transfer End of Path** or **Abandoned Out of Hours**).

DNIS Call Type Exit Point

This path shows how sessions are tracking from each dialed number by call-type, through to the exit point of the call (the block where the caller hung up).

CallType Disposition Path

This path shows how the sessions are tracking for each call type, through the high-level disposition categories to a final disposition.

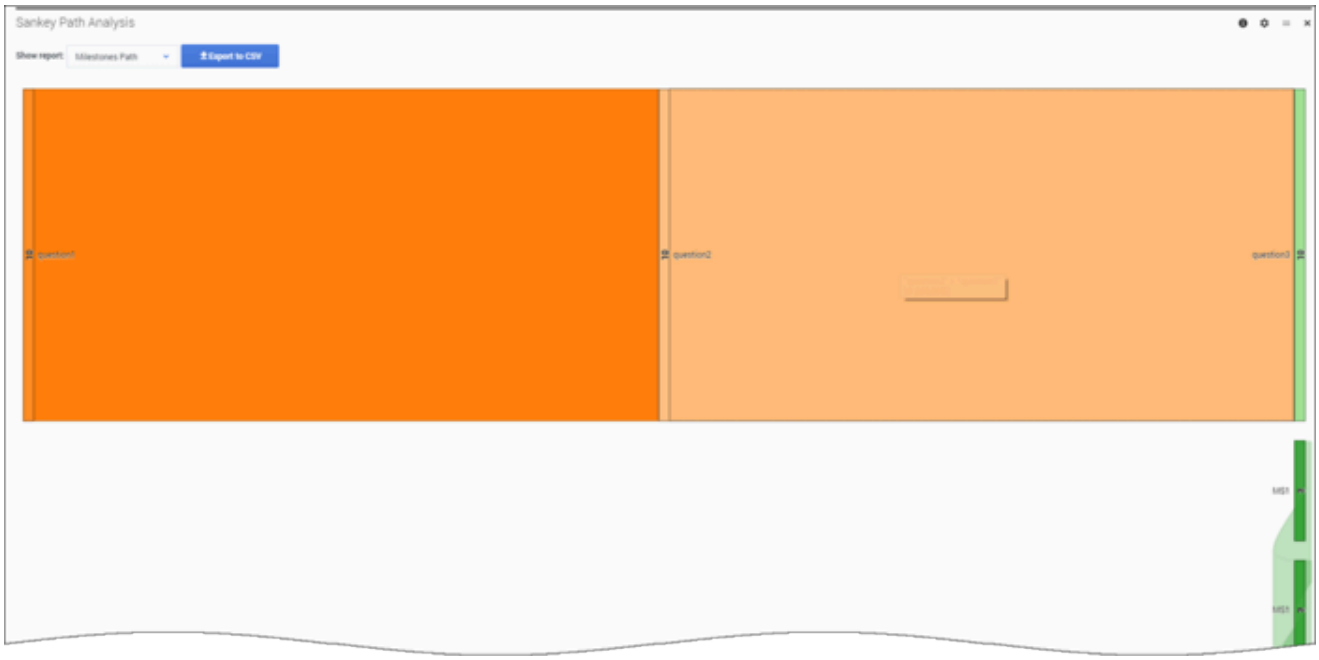
Call Type Disposition Exit Point

This path shows how the sessions are tracking by call type, through the high-level disposition categories to the exit point of the call (the block where the caller hung up).

Sankey Path Analysis

This dashboard is similar to the Path dashboard, except that it generates the results based on milestones and menu inputs.

> **Tip**
>
> The Sankey panel might not be able to render correctly if there are several nodes or links to display. When this happens, you'll be prompted to increase the height of the panel.



Reports on this dashboard

Count Over Time

(See Count Over Time.)

Filter

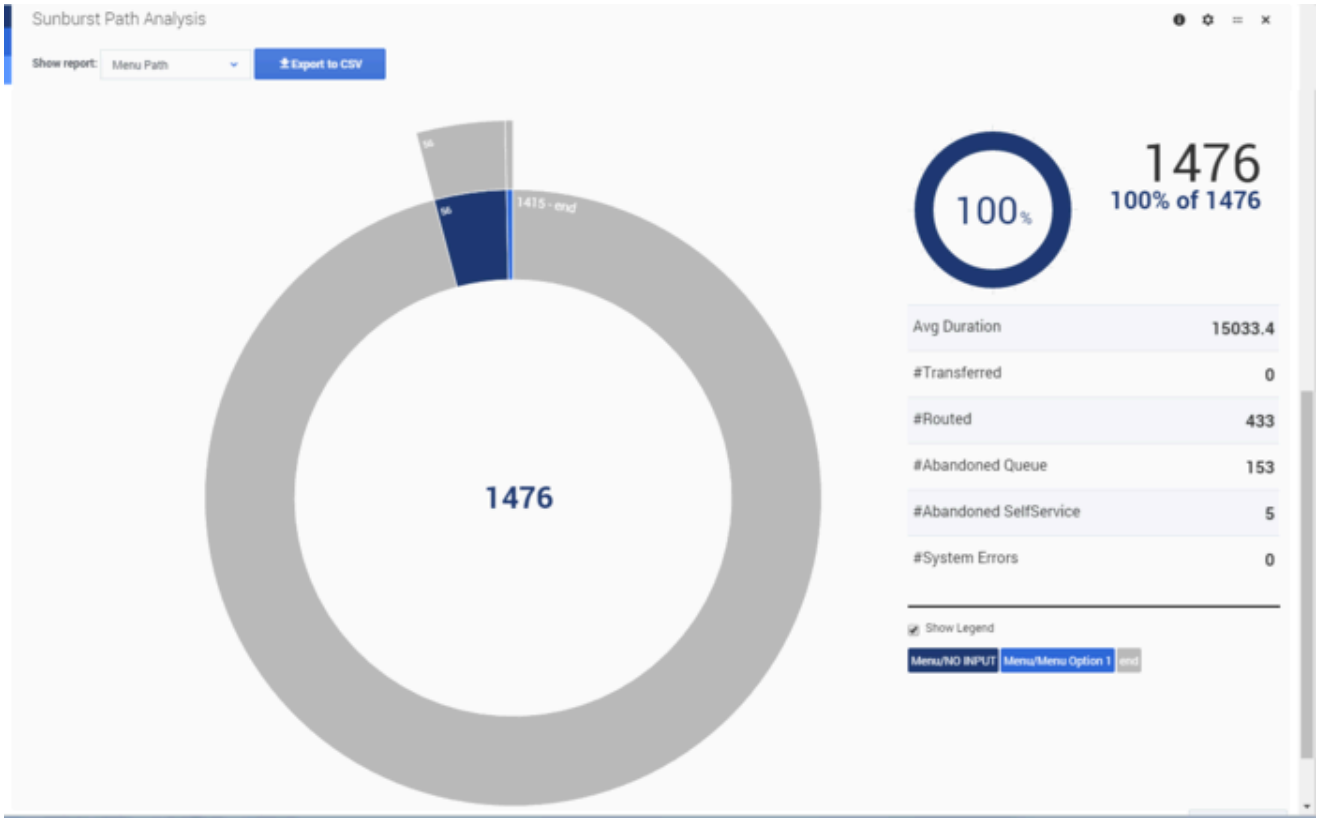
(See Filter.)

Sankey Path Analysis

This diagram shows how sessions are tracking through a series of milestones or menu inputs. You can toggle between the different reports using the **Show report** menu, and export the results to a CSV file.

Sunburst Path Analysis

The **Sunburst Path Analysis** dashboard shows a visual representation of the menu and milestone paths for your application sessions.



Reports on this dashboard

Count Over Time

(See Count Over Time.)

Filter

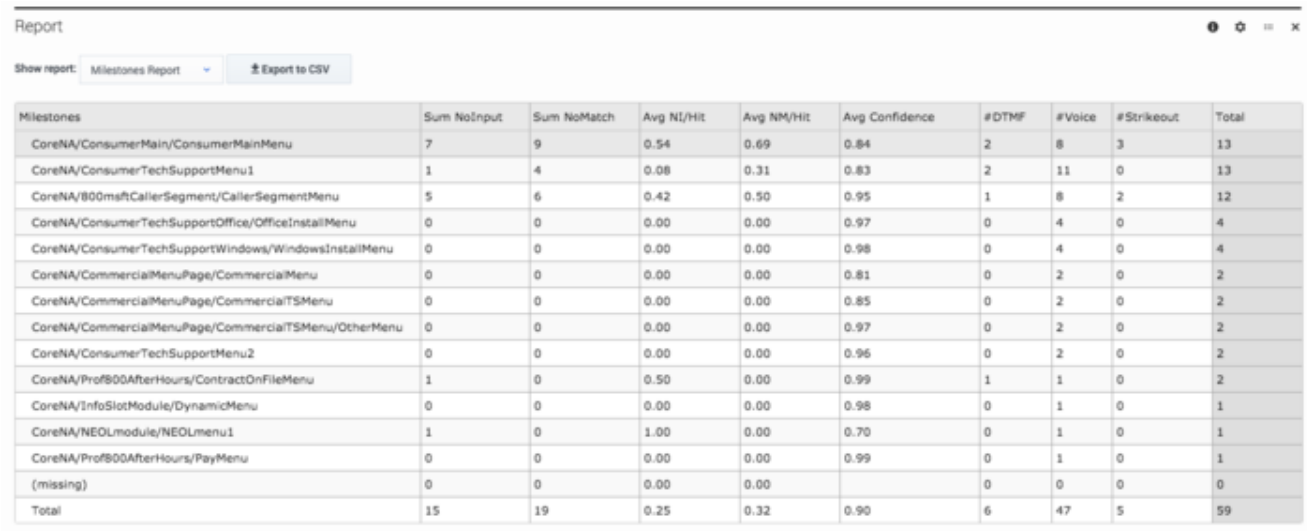
(See Filter.)

Sunburst Path Analysis

By default, this report displays the first seven nodes of a menu or milestones path, but you can change this in the configuration settings for the panel. The center of circle shows the total count, and you can double-click any of the partitions to drill down for more information.

Inputs

The **Inputs** dashboard shows you information about the menu inputs for application sessions that were active during the given time window. For example, you can see the number of sessions where there was **No Input**, **No Match**, or a **Strikeout**.


Report

Show report: Milestones Report ▾ Export to CSV

Milestones	Sum NoInput	Sum NoMatch	Avg NI/Hit	Avg NM/Hit	Avg Confidence	#DTMF	#Voice	#Strikeout	Total
CoreNA/ConsumerMain/ConsumerMainMenu	7	9	0.54	0.69	0.84	2	8	3	13
CoreNA/ConsumerTechSupportMenu1	1	4	0.08	0.31	0.83	2	11	0	13
CoreNA/800msftCallerSegment/CallerSegmentMenu	5	6	0.42	0.50	0.95	1	8	2	12
CoreNA/ConsumerTechSupportOffice/OfficeInstallMenu	0	0	0.00	0.00	0.97	0	4	0	4
CoreNA/ConsumerTechSupportWindows/WindowsInstallMenu	0	0	0.00	0.00	0.98	0	4	0	4
CoreNA/CommercialMenuPage/CommercialMenu	0	0	0.00	0.00	0.81	0	2	0	2
CoreNA/CommercialMenuPage/CommercialTSMenu	0	0	0.00	0.00	0.85	0	2	0	2
CoreNA/CommercialMenuPage/CommercialTSMenu/OtherMenu	0	0	0.00	0.00	0.97	0	2	0	2
CoreNA/ConsumerTechSupportMenu2	0	0	0.00	0.00	0.96	0	2	0	2
CoreNA/Prof800AfterHours/ContractOnFileMenu	1	0	0.50	0.00	0.99	1	1	0	2
CoreNA/InfoSlotModule/DynamicMenu	0	0	0.00	0.00	0.98	0	1	0	1
CoreNA/NEOLmodule/NEOLmenu1	1	0	1.00	0.00	0.70	0	1	0	1
CoreNA/Prof800AfterHours/PayMenu	0	0	0.00	0.00	0.99	0	1	0	1
(missing)	0	0	0.00	0.00		0	0	0	0
Total	15	19	0.25	0.32	0.90	6	47	5	59

Reports on this dashboard

No Input

The total number of sessions with **No Input**.

No Match

The total number of sessions with **No Match**.

Strikeout

The total number of sessions where the maximum number of inputs for **No Input** or **No Match** was reached.

Report

This report lists the milestones and their various input counts. You can use **Show Report** to toggle the following report views:

- Milestones Report
- Milestones and Utterances Report
- Milestones and Interpretations

You can also export the results to a CSV file.

Report by Block

This report lists the blocks and their various input counts. You can use **Show Report** to toggle the following report views:

- Blocks and Utterances Report
- Blocks and Interpretations Report

You can also export the results to a CSV file.

> ### Tip
>
> An **utterance** is what Designer believes the caller has *said*. An **interpretation** is what Designer thinks the caller actually *meant* (in a context that is meaningful to the application).
>
> For example, a caller might say "I need to speak with *someone.*" This becomes an utterance value that Designer could map to an interpretation value of "I need to speak with an *agent*", which enables the application to respond appropriately.

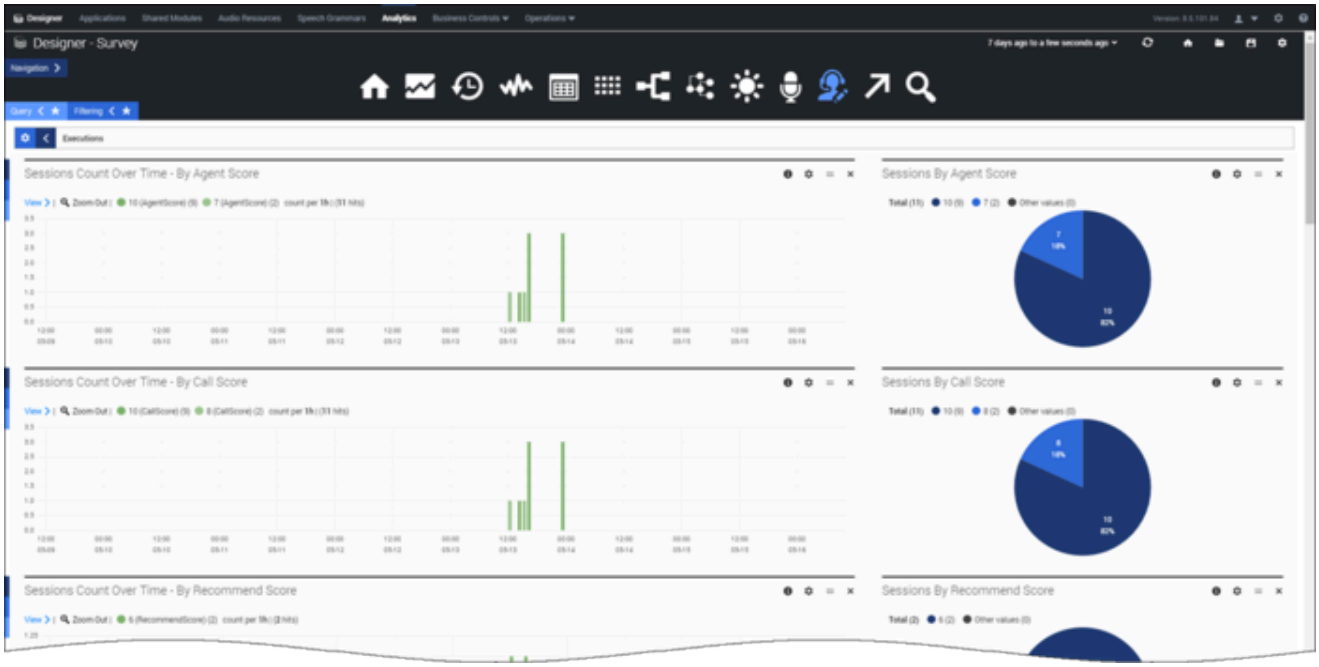
Blocks By NoMatch/NoInput Count

This report shows the number of times an input field encounters a No Match or No Input per call.

Use **Show Report** to toggle the **Blocks By NM/NI Count** report. You can also export the results to a CSV file.

Surveys

The **Surveys** dashboard gives you information related to your survey applications.



The **Surveys** dashboard contains many of the same panel types that are shown on other dashboards, but also includes data that is specific to the surveys in your applications (mostly pulled from the variables you initialized when the survey was set up).

Reports on this dashboard

Survey Count Over Time

Similar to Count Over Time, but specific to the number of surveys that were offered, accepted, or rejected.

Filter

(See Filter.)

Sessions Count Over Time

These histogram reports break the number of session counts down by:

- **Product Score**
- **Company Score**
- **Recommend Score**

These are based on the ratings callers gave during the survey.

Sessions By Score

These pie graph reports are in pie graph format and break the number of session counts down by:

- **Product Score**
- **Company Score**
- **Recommend Score**

These are based on the ratings callers gave during the survey.

Survey Answer Distribution

This report lists the questions that were presented to callers, along with the average scores for each question, the total number of sessions in which the question was presented, and the total number of sessions where the question was answered.

Sunburst Path Analysis

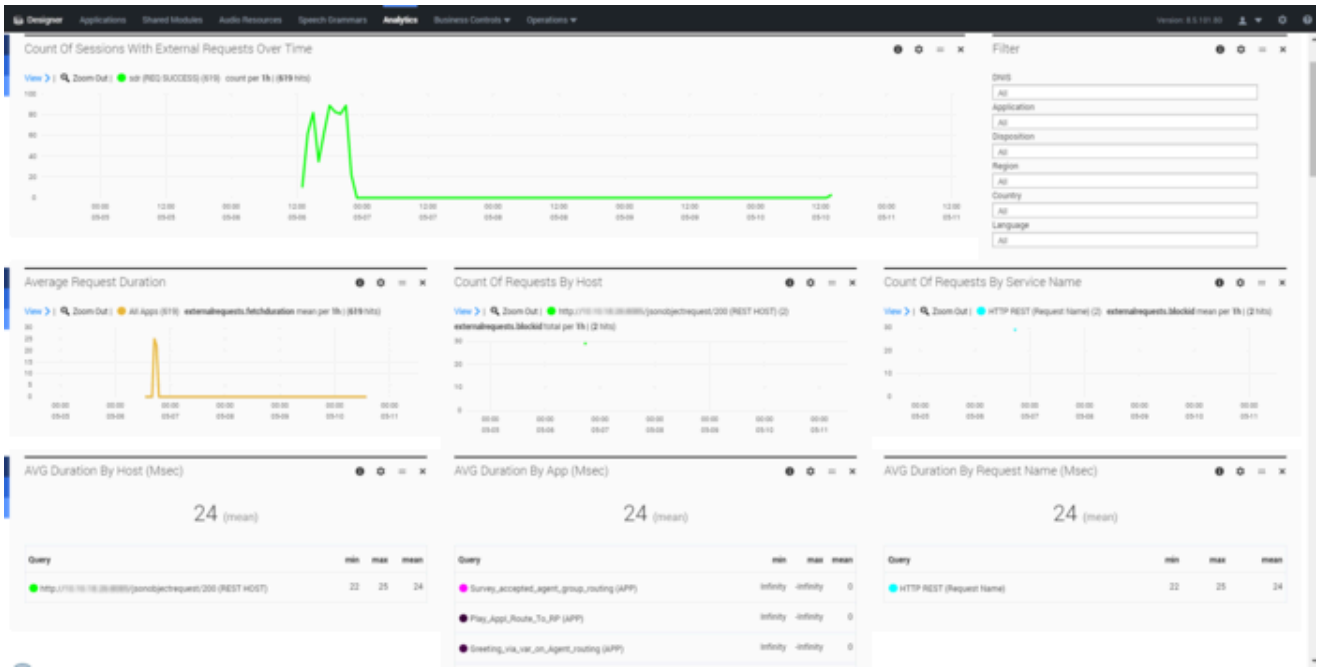
This report gives you a quick snapshot of how many surveys were offered, accepted, or rejected.

External Services

Designer applications can request a connection to a web-accessible external system, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM), to fetch or update data about a call. These integration points between applications and external services can play an important role, such as determining whether a call stays within **Self Service** or is routed to an agent through **Assisted Service**.

Currently, there are two blocks that can make these types of requests: Custom Service and HTTP REST.

The reports on this dashboard will help you to analyze the external requests being made by your applications.



Reports on this dashboard

Count of Sessions With External Requests Over Time

This report captures the number of external service requests that were made by the applications during the given time period. You can quickly see how the number of requests compares to the traffic patterns, and spot any trends or unexpected deviations.

For example, if you see that web service requests are increasing during certain times, you can check to make sure that the external services can handle that volume of requests.

Filter

(See Filter.)

Average Request Duration

You can use this report to see how long (on average) it is taking for the external services to respond to requests from the applications.

For example, if the response time trajectory is flat, it typically means that data is being retrieved or updated as expected. But if the response times are increasing significantly as your call volumes rise (it's normal to see a slight increase in response times whenever there is a spike in call volumes), it might indicate that the external system is becoming strained while trying to handle so many requests at the same time.

Count Of Requests By Host and Service Name

These reports provide the number of external service requests (or hits) for each Host of a particular Service Name.

- *Host* is the domain name of the URL that is receiving a request from the HTTP REST or Custom Service block.
- *Service Name* is the name of the block used within the application.

You can use this information to identify which hosts or services are receiving the most hits, and then plan the external requests accordingly. If you are using third party integrations (such as payment gateways or location services), this data can provide insight into the consumption of those types of services.

AVG Durations By Host, App, and Request Name

These reports tell you how long (on average) it is taking the external web services to respond to requests. This average is calculated by adding the response times of all service requests and dividing them by the number of requests.

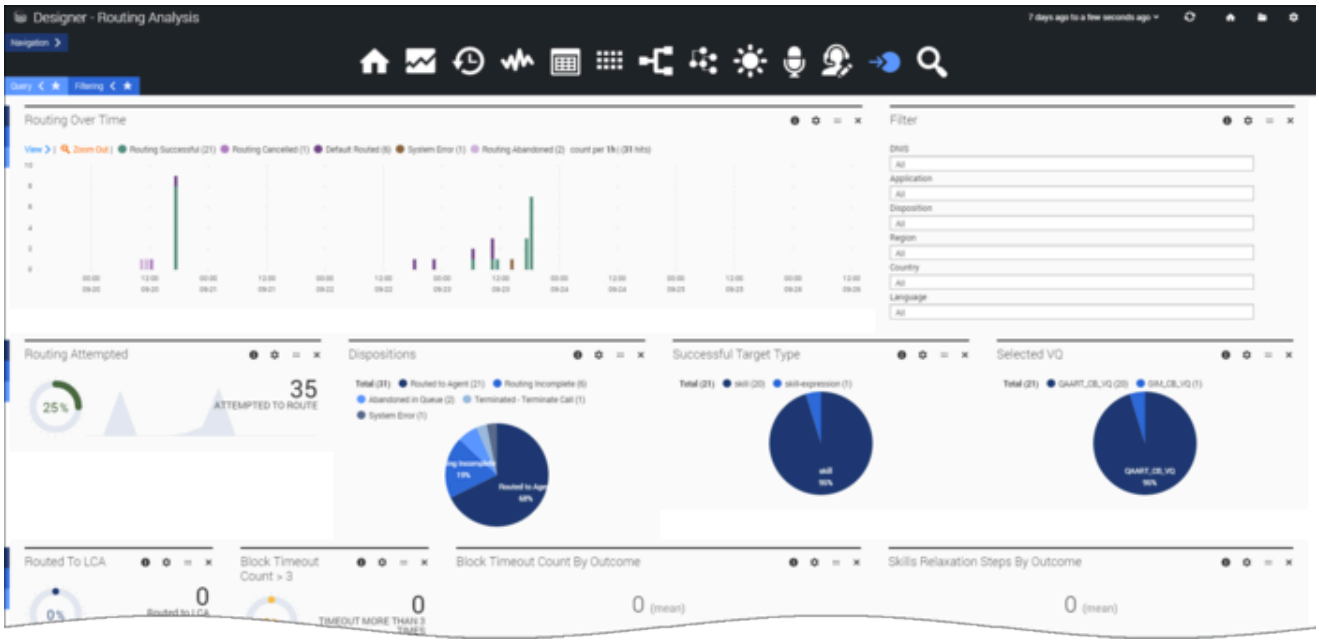
Analyzing the average durations based on host, URL, or application can help you identify if services are responding within the expected timeframe.

External Requests Status

This report shows the status of all external requests that were made during the given time period.

Routing Analysis

This dashboard gives you a deeper look into those sessions that have entered the Assisted Service phase of the applications.



These reports can help to answer questions like:

- How many sessions were successfully routed to their destination?
- How long are callers waiting in the queue before hanging up?
- Which applications had the lowest abandoned rates?
- Which blocks had the least number of timeouts?

Reports on this dashboard

Routing Over Time

The total number of routing sessions that happened during the given time period, broken down by status (such as, if the routing session was abandoned, successful, cancelled, default routed, or had an error).

Filter

(See the Summary dashboard for a description of this panel.)

Routing Attempted

The total count and percentage of sessions that entered the routing phase (based on the number of all sessions processed during the given time period).

Dispositions

The total number of routing attempts made, broken down by disposition.

Successful Target Type

The number of sessions that were successfully routed to the target, broken down by target type (such as a specific skill).

Selected VQ

The number of sessions that were successfully routed to the selected virtual queue.

Routed to LCA

The number of sessions that were successfully routed to the last called agent (that is, the agent that the customer last spoke with).

Timeout Block Count > 3

The total count and percentage of all routing sessions where the routing blocks had three (3) or more timeouts.

Block Timeout Count by Outcome

The total count, including the minimum, maximum, and mean, of routing block timeouts experienced by routing sessions, broken down by routing outcome (positive/negative).

Skills Relaxation Steps by Outcome

The total count, including the minimum, maximum, and mean, of skill relaxation steps, broken down by routing outcome (positive/negative).

Average Time in Queue (in seconds)

The average amount of time that sessions waited in the queue before being routed.

Abandoned in Queue Stats

Details about routing sessions that were abandoned while still in the queue, such as how long callers waited before hanging up.

Performance Report by Applications

This table shows the performance breakdown by individual applications, as based on a variety of metrics. This panel lets you see which applications have the best (or worst) performance.

Performance Report by Blocks within Applications

This table shows the performance breakdown by individual blocks, as based on a variety of metrics. This panel lets you see which blocks have the best (or worst) performance for any given application.

Skills Path

This panel provides a sankey visualization of the skills path. (By default, the depth of this panel is set to 7.)

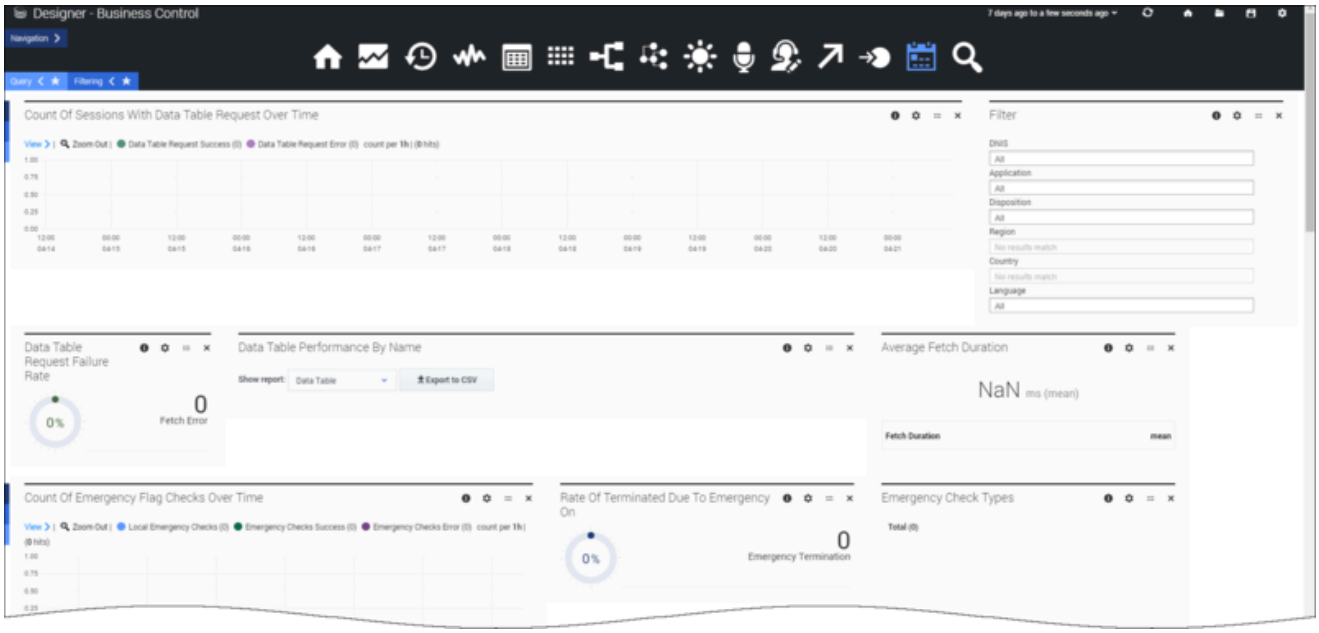
Abandoned in Queue Stats

This panel provides a sunburst visualization of the Abandoned in Queue stats. *[BM - Not sure if this report will make the release; there is a note in the design spec that indicates a formatting issue with it.]*

Business Control Dashboard

The Business Control dashboard provides near real-time metrics on the runtime usage of Emergency Flags, Business Hours, Special Days, and Data Tables.

> **Important**
>
> To populate the reports on this dashboard with data, you must first set the value of the **SdrTraceLevel** system variable in the application to `100` (debug level). For more information, see System Variables.



Reports on this dashboard

> **Tip**
>
> If a business control object that was checked in a session is deleted, the report adds the suffix `(deleted)` to the object ID.

Filter

(See the Application Details dashboard page for a description of this panel.)

Data Tables

Count of Sessions With Data Table Request Over Time

This report shows you the number of data table retrieval requests that were received over a period of time. You can quickly adjust the range of time shown, say, for the last hour, 24 hours, or the last week, to see the trends for those specific time periods.

Each bar indicates the total number of data table retrieval requests for each application that took place during the given time period. The higher the bar, the higher the number of requests. You can easily see what time of the day (or what day of the week) that requests are higher or lower, and organize your business accordingly.

Data Table Request Failure Rate

The failure rate (as a percentage) of data table requests across all sessions. It also shows the total number of Fetch Errors.

Data Table Performance by Name

A breakdown of outcomes for the most requested data tables. You can export the results to a CSV file.

Data Table Request Fetch Duration

The average duration (ms) of data table fetch requests, grouped by outcome (success or failure).

Emergency Flags

Count of Emergency Flag Checks Over Time

The total number of Emergency Flag checks that were sent during the selected period of time.

Rate of Terminated Due to Emergency On

The total rates of sessions (where an emergency flag was checked) that were terminated due to the Emergency flag being on.

Emergency Check Types

A breakdown of Emergency checks by type.

Emergency Check Error Rate

The rate of errors (as a percentage) across all external emergency checks. It also shows the total of Check Errors.

Shared Emergency Flag Performance

A breakdown of outcomes for the most checked Emergency flags. You can export the results to a CSV file.

External Emergency Check Fetch Duration

The average duration (ms) of Emergency check fetch requests, grouped by outcome (success or failure).

Business Hours

Count of Business Hours Checks Over Time

The total number of Business Hours checks that were sent during the selected period of time.

Rate of Terminated Due to Business Closed

The total rates of sessions (where Business Hours were checked) that were terminated due to the business being closed.

Business Hours Check Types

A breakdown of Business Hours checks by type.

Business Hours Check Error Rate

The rate of errors (as a percentage) across all external Business Hours checks. It also shows the total of Check Errors.

Shared Business Hours Check Performance

A breakdown of outcomes for the most checked Business Hours. You can export the results to a CSV file.

Business Hours Check Fetch Duration

The average duration (ms) of Business Hours check fetch requests, grouped by outcome (success or failure).

Special Days

Count of Special Days Checks Over Time

The total number of Special Days checks that were sent during the selected period of time.

Rate of Terminated Due to Special Days

The rate of sessions (where Special Days were checked) that were terminated due to it being a

Special Day.

Special Days Hours Check Error Rate

The rate of errors (as a percentage) across all external Special Days checks. It also shows the total of Check Errors.

Special Days Check Type Breakdown

A breakdown of Special Days checks by type.

Shared Special Days Performance

A breakdown of outcomes for the most checked Special Days. You can export the results to a CSV file.

Special Days Check Fetch Duration

The average duration (ms) of Special Days check fetch requests, grouped by outcome (success or failure).

Top Hit Holidays

A list of the top holidays where a Special Days check returned *true*.

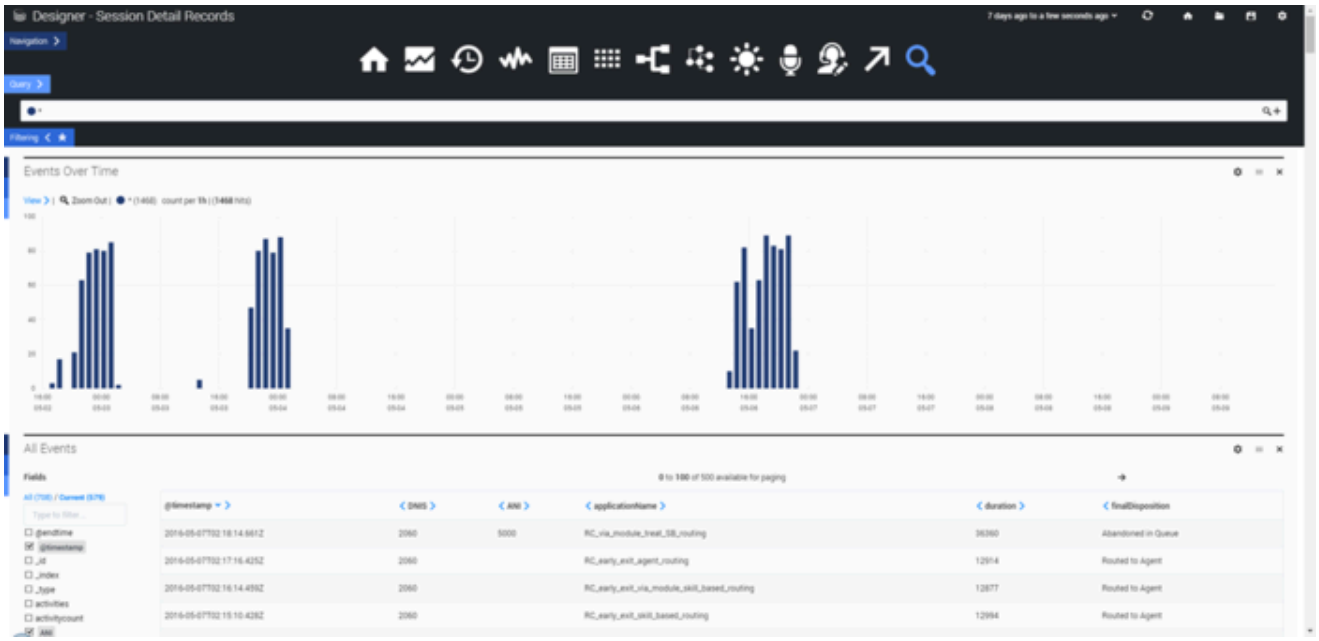
Errors

Count of Sessions With Business Control Errors Over Time

The total counts of sessions where errors in Data Table, (shared) Emergency Flag, (shared) Special Days and (shared) Business Hours checks were encountered, over the selected period of time.

Session Detail Records

The **Session Detail Records** dashboard lets you view (and query) some of the raw data contained in the Session Detail Records (SDRs).



Reports on this dashboard

Events Over Time

Similar to Count Over Time, this report shows the number of events that were logged during the given time period.

All Events

Basically, this is a table showing raw data information for all application sessions that were active during the given time period. You can use the **Fields** check boxes to select which columns to display. You can then search the results for the selected columns by toggling the **Query** option and entering your own query.

Here's an example of a query statement:

```
=applicationName=Test 2 AND finalDisposition=System Error AND ANI=703-123-1234
```

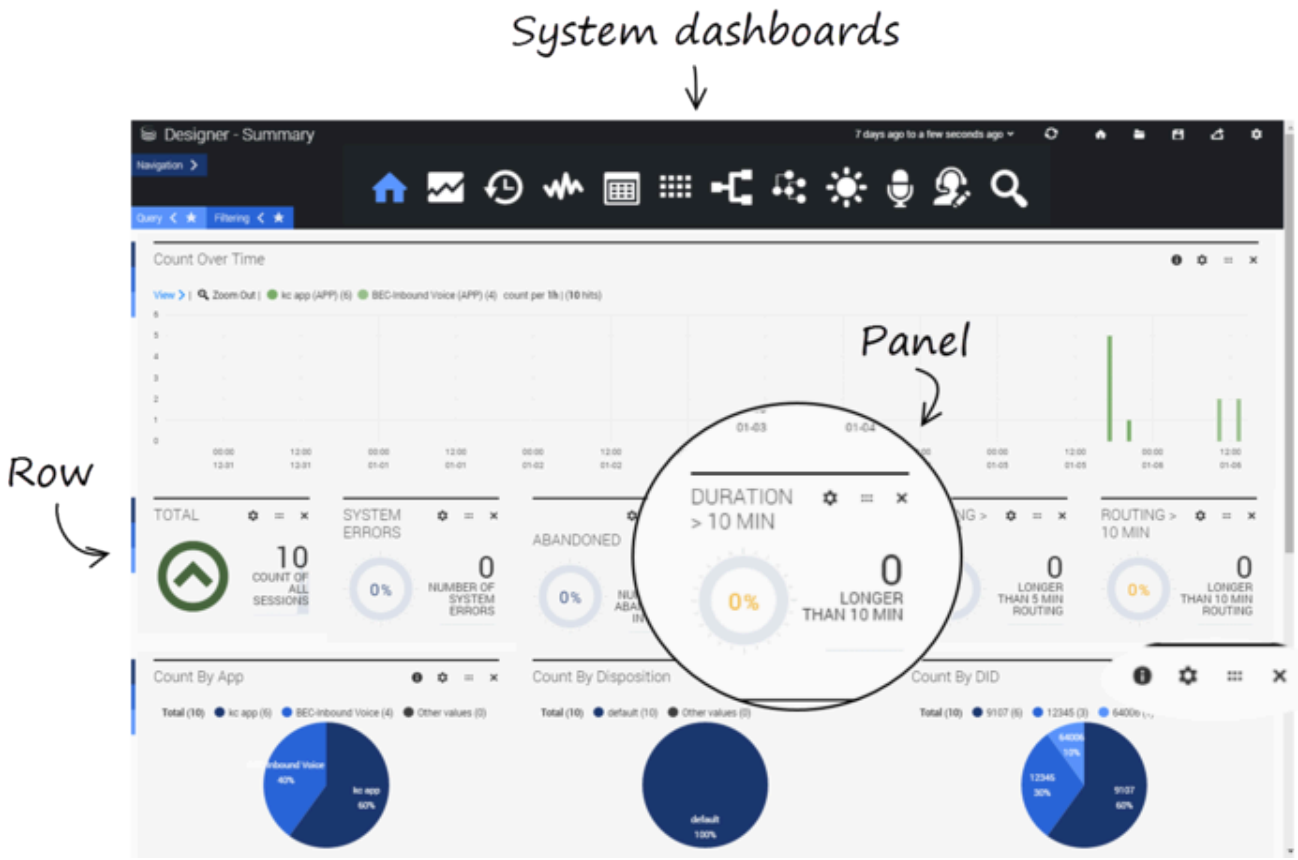
This query would search the **Test 2** application for sessions where the final disposition was **System**

Error and the ANI was **703-123-1234**.

After the results are generated, click an event to expand it. You can then view its details in **Table**, **JSON**, or **Raw** format. When an event is expanded, you can also use the **Action** menu to add/remove the item from the filter, or to add/remove that column from the table.

Dashboard management

Each dashboard is made up of several rows, each containing one or more reporting panels. The buttons on the main dashboard toolbar let you quickly navigate between the various types of system dashboards.



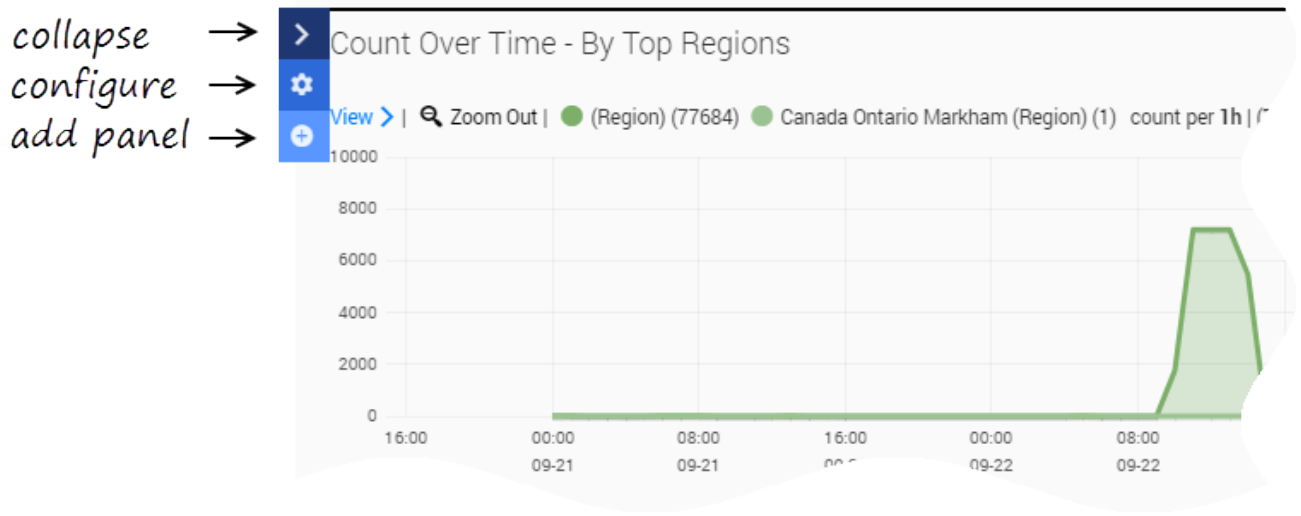
The settings for the built-in system dashboards cannot be changed.

Tip

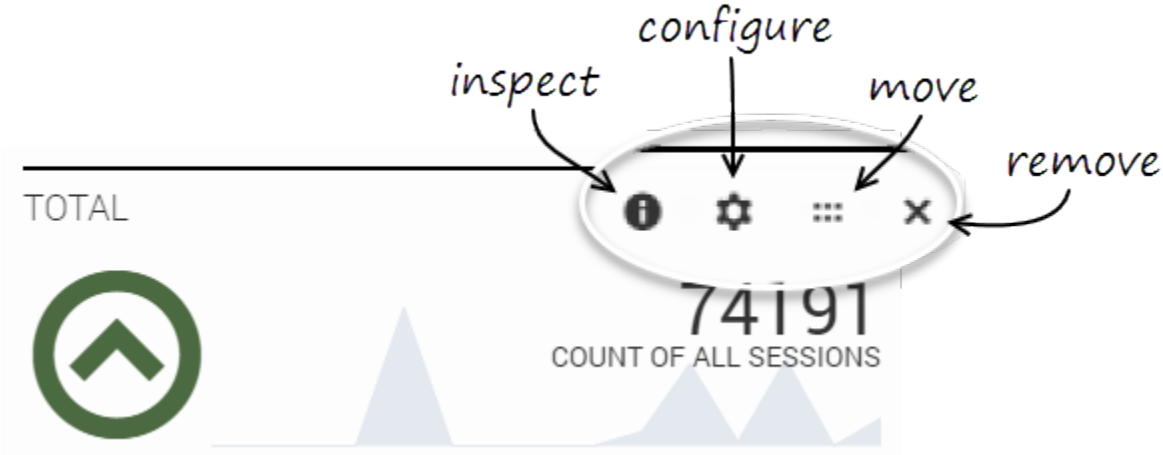
The Designer Dashboards are built on the Kibana plug-in (by elastic). For additional details that are not covered in this Help, please refer to the Kibana 3.0 documentation.

Rows and Panels

On the left side of each row is a sliding menu that remains hidden (or collapsed) when not in use. The menu slides out when you hover over it and has options to **collapse**, **configure**, or **add panels** to the row.



Each of the panels has options to **configure**, **move**, or **remove** the panel. Some panels also have an **inspect** icon that lets you view the query for the report being displayed.

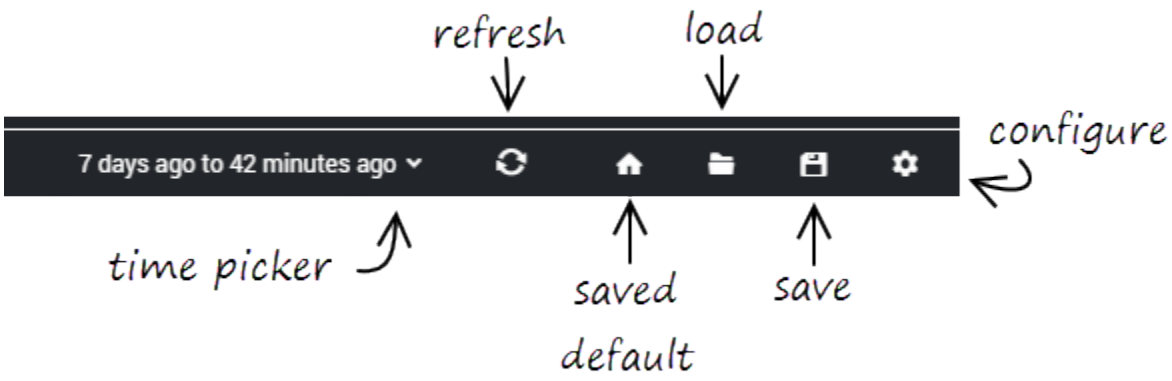


Tip

Sometimes, after closing an **inspect** panel, it might look like the dashboard page has gone blank, especially if there was a lot of information to scroll through. If this happens, simply scroll back up to the top of the dashboard page. (This issue can be avoided by using **Esc** to dismiss an **inspect** panel instead of closing it.)

Dashboard controls

At the top of the screen are control icons for performing common tasks.



Time picker

Lets you select the time period for which you want to display data.

Refresh

Refreshes the dashboard with the most current data for the selected time period.

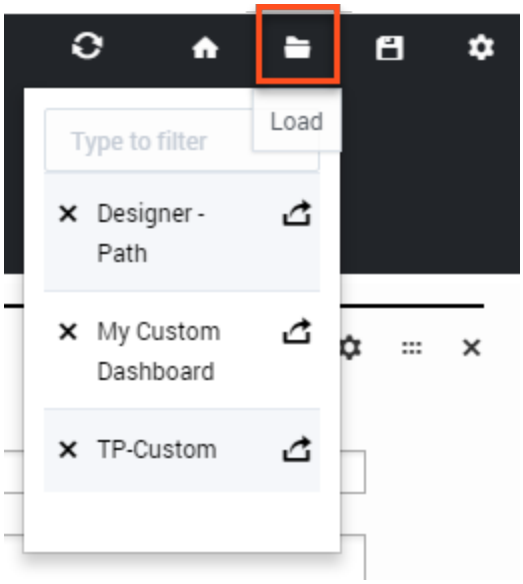
Home (saved default)

Returns to the dashboard that is currently saved as the default (or "home") dashboard.

> **Tip**
> This **Home** button is not the same as the **Home** button on the main toolbar (for the built-in dashboard types). That button will always return you to the built-in Summary dashboard, while this one will return you to the dashboard you've assigned to it.

Load

Lets you select and open any dashboards that you have saved.



Save

Saves the current dashboard. You will be prompted to give the dashboard a new name. (This option might not be available for all users or deployments.)

Configure

Customize certain properties and settings for the current dashboard.

Navigation, Query, and Filtering tabs

Toggling these tabs lets you show or hide their options.

Navigation tab

Shows or hides the dashboard icons.

Query tab

Toggling the **Query** tab lets you manually enter a query statement to search for specific terms, criteria, or conditions.

For example, you might enter the following query:

```
=applicationName=Test 2 AND finalDisposition=System Error AND ANI=703-123-1234
```

This query would search the **Test 2** application for sessions where the final disposition was **System Error** and the ANI (the caller's phone number) was **703-123-1234**.

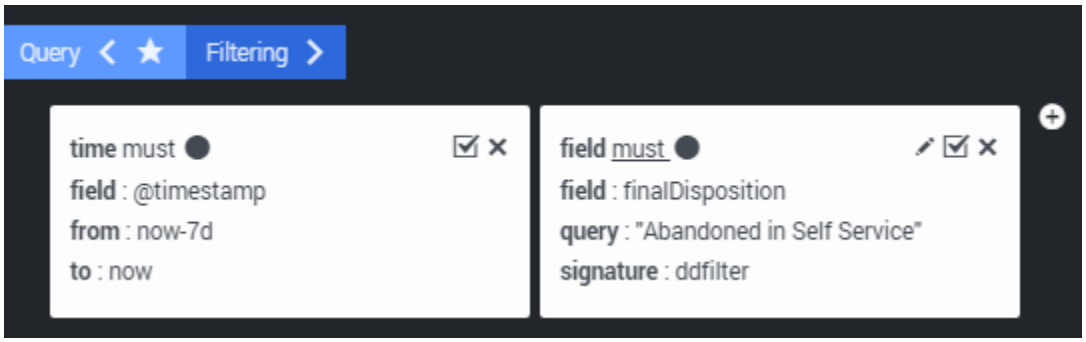
Query statements are not automatically applied to all panels on the dashboard. To use a query statement on a panel, go to the panel's **Configure** settings and select it from the **Queries** tab.

Filtering tab

Toggling the **Filtering** tab lets you see any filters that are currently applied to the dashboards.

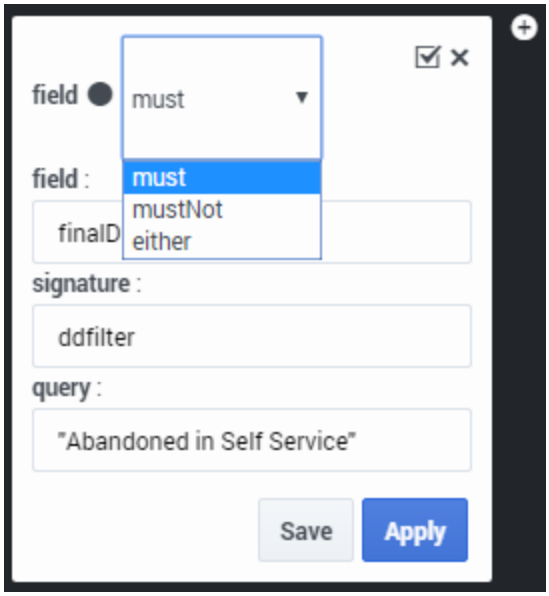
> ### Important
> In most cases, any filters that you define are applied across *all* of the system dashboards, not just the dashboard you are currently viewing.

Whenever you select a value from the Filter panel, or click a specific time range or section of a pie or bar chart, a filter is automatically created. In this example, we can see that an **Abandoned in Self Service** filter was applied:



With the filters visible, you can easily modify, remove, or turn a filter on or off (the solid dot indicates the filter is currently *on*).

You can also make a filter *inclusive* or *exclusive* by clicking the field and choosing one of the options:


field
must
field :
must
mustNot
either
finalD
signature :
ddfilter
query :
"Abandoned in Self Service"
Save
Apply

Session Detail Record (SDR) Fields Reference

This page lists some of the most commonly used Session Detail Record (SDR) fields. It is not intended as a comprehensive list of all SDR fields used by Designer.

In most cases, these fields are automatically populated with system-generated values. But in some cases, the values are based on information provided by the application developer, such as the name of the country and the language of the application.

The SDR fields are organized according to the block category (or classification) they are most typically associated with.

> **Tip**
> Not sure which category your field belongs to? Use the search function in your browser to see if it's listed on this page. For example, in Chrome, press **CTRL + F** to open the search tool.

Session Data

The following SDR fields contain values that are sourced from data collected during various phases of the application session. They are mostly used for generating reports on the Designer Dashboards but can also be used for troubleshooting.

Values set by Designer

Field	Description
`@endtime`	Timestamp of when the interaction ended. Example: `2017-03-08T01:56:26.085Z`
`@timestamp`	Timestamp of when the interaction started. Example: `2017-03-08T01:56:12.037Z`
`ANI`	The caller's phone number (or Caller ID).
`asduration`	Length of time that the call was in the Assisted Service phase.
`asend`	Timestamp of when the Assisted Service (or IVR) phase of the interaction ended. If the interaction was routed, this value represents the time that the session exited the Self Service phase.

Field	Description
	Example: 2017-03-08T01:56:20.937Z
asstart	Timestamp of when the interaction entered the Assisted Service phase. Example: 2017-03-08T01:56:20.933Z
callEndReason	Indicates why a call ended, such as whether it was terminated in Self Service, Assisted Service, or due to a System Error.
DNIS	The phone number that the caller dialed.
duration	Duration of the total session, in milliseconds (ms). **Tip:** To calculate the duration of blocks execution and exclude session wrap-up time, subtract the value of the operationalOverheadDuration field from this value.
operationalOverheadDuration	Total time that the application was in an idle state (typically, this occurs just before the session finalizes).
finalDisposition	Status of an interaction at the time it exited the application flow, such as whether it was routed to an agent or the caller hung up. (See Count by Disposition for more information about dispositions.)
InteractionCategory	Channel type used for the interaction (voice, chat, or email).
InteractionID	Unique ID of the interaction. This ID can be used to track an interaction across multiple application sessions (or multiple instances of the *same* application).
ssduration	Amount of time that the caller spent in the Self Service phase of the application (in milliseconds).
ssend	Timestamp of when the Self Service (or IVR) portion of the interaction ended.
ssstart	Timestamp of when the Self Service phase of the application started.

Values set by application developer

These fields are populated with values that were provided by the application developer.

Field	Description
activities	List of activities encountered in the interaction. Activities are defined in an Activity block or automatically captured when an interaction enters or exits a shared module.
activitycount	Number of activities that were referenced in the interaction.
CountryName	Name of the country.
LanguageName	Language of the application.

Business Controls

These fields contain values sourced from Busines Controls blocks.

Field	Description
businesshourserrcount	Number of times a Business Hours block was accessed during the interaction.
businesshoursextcount	Total number of external business hours checks that occurred within a session.
businesshoursreqcount	Total number of requests for business hours checks that occurred within a session.
emergencieserrcount	Total number of emergency flag checks within a session.
emergenciesextcount	Total number of external emergency flag checks within a session.
specialdayserrcount	Total number of special days checked within a session (all checks in a single Special Day block count as one check).
specialdaysextcount	Total number of special days checked externally within a session (all checks in a single Special Day block count as one check).
specialdaysreqcount	Total number of exceptions encountered when checking special days (all errors in a single Special Day block count as one check).

Routing Data

These fields contain values sourced from Routing blocks.

Field	Description
routingBlockCount	Number of routing blocks that were hit within a session.
routingBlockTimeoutCount	Number of times that routing blocks timed out. Tip A high number here can indicate that customers are waiting too long in the queue for some reason. For example, skill levels might be set too high or there are not enough agents available.
routingCallHandlingType	Indicates the type of routing used for the call (default or consult).
routingLCAAttempted	Indicates if **Last Called Agent** routing was attempted (this option can be enabled on the Route Agent block).
routingSkillRelaxationCount	Number of routing blocks that used skill relaxation.

User Interaction

These fields contain values sourced from User Interaction blocks.

Field	Description
inputcount	Total number of User Input blocks executed during a session.
nicount	Total count of **No Input** instances that occurred for Input class blocks (User Input and Menu).
nmcount	Total count of **No Match** instances that occurred for Input class blocks (User Input and Menu).

Field	Description
strikeoutcount	Count of inputs where the maximum number of permitted attempts was exceeded.

External Requests

These fields contain values sourced from External Services blocks.

Field	Description
extreqcount	Total number of external requests.
extreqerrorcount	Total number of failed external requests.

User Milestone

These fields contain values sourced from user-defined milestones (such as those defined in a Milestone block or Menu block) and Survey blocks.

Field	Description
FinalUserMilestone	The last milestone in the milestones array at the end of the application session.
LastMilestone	Tracks the most recent milestone encountered.
milestones	List containing the system milestones that were encountered. Milestones indicate special points or transitions in the application, such as phases starting, phases ending, or an application terminating. **Note:** These values can be auto-populated by Designer or provided by the application developer.
userMilestones	Milestones that were defined in Milestone blocks or set in other blocks, such as Menu. **Note:** These values can be auto-populated by Designer or provided by the application developer.
usermilestonecount	Number of user-defined milestones hit in the session.
userMilestonesPath	Names of all milestones hit during the call.

Activity

These fields contain values sourced from Activity blocks.

Field	Description
activities	List of activities encountered in the interaction. Activities are defined in an Activity block or automatically captured when an interaction enters or exits a

Field	Description
	shared module.
activitycount	Number of activities that were referenced in the interaction.