Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Started</td>
<td></td>
</tr>
<tr>
<td>Getting Started with Designer</td>
<td>7</td>
</tr>
<tr>
<td>Quick links</td>
<td>16</td>
</tr>
<tr>
<td>Permissions and Access</td>
<td>19</td>
</tr>
<tr>
<td>Video Gallery for Designer</td>
<td>24</td>
</tr>
<tr>
<td>Upgrade notes</td>
<td>30</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
</tr>
<tr>
<td>Applications</td>
<td>33</td>
</tr>
<tr>
<td>Application Settings</td>
<td>51</td>
</tr>
<tr>
<td>Application phases</td>
<td>66</td>
</tr>
<tr>
<td>About application resources</td>
<td>74</td>
</tr>
<tr>
<td>Using the blocks</td>
<td>83</td>
</tr>
<tr>
<td>Application workflow</td>
<td>98</td>
</tr>
<tr>
<td>Variables</td>
<td>105</td>
</tr>
<tr>
<td>Saving and Publishing Your Application</td>
<td>119</td>
</tr>
<tr>
<td>Build a basic application</td>
<td></td>
</tr>
<tr>
<td>Shared Modules</td>
<td></td>
</tr>
<tr>
<td>Shared Modules</td>
<td>123</td>
</tr>
<tr>
<td>Media Resources</td>
<td></td>
</tr>
<tr>
<td>Media Collections</td>
<td>129</td>
</tr>
<tr>
<td>Personas</td>
<td>142</td>
</tr>
<tr>
<td>Intelligent prompts</td>
<td>147</td>
</tr>
<tr>
<td>Digital Resources</td>
<td></td>
</tr>
<tr>
<td>Digital Resources</td>
<td>152</td>
</tr>
<tr>
<td>Manage personally identifiable information</td>
<td>155</td>
</tr>
<tr>
<td>Manage standard responses</td>
<td>163</td>
</tr>
<tr>
<td>Manage field codes</td>
<td>175</td>
</tr>
<tr>
<td>Manage custom variables</td>
<td>184</td>
</tr>
<tr>
<td>Bots and Speech Grammar Resources</td>
<td></td>
</tr>
<tr>
<td>Speech Grammars</td>
<td>188</td>
</tr>
<tr>
<td>Bot Registry</td>
<td>191</td>
</tr>
<tr>
<td>Business Controls Resources</td>
<td></td>
</tr>
<tr>
<td>Business Controls</td>
<td>198</td>
</tr>
<tr>
<td>Emergency Flags</td>
<td>201</td>
</tr>
<tr>
<td>Special Days</td>
<td>204</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Business Hours</td>
<td>206</td>
</tr>
<tr>
<td>Data Tables</td>
<td>210</td>
</tr>
<tr>
<td><strong>Admin Resources</strong></td>
<td></td>
</tr>
<tr>
<td>Admin Resources</td>
<td>228</td>
</tr>
<tr>
<td>Partition-Based Access Control</td>
<td>230</td>
</tr>
<tr>
<td><strong>Logic and Control Blocks</strong></td>
<td></td>
</tr>
<tr>
<td>Logic and Control Blocks</td>
<td>237</td>
</tr>
<tr>
<td>Assign Variables Block</td>
<td>240</td>
</tr>
<tr>
<td>Change Language Block</td>
<td>245</td>
</tr>
<tr>
<td>Classify Block</td>
<td>249</td>
</tr>
<tr>
<td>Click-to-Call-In Match Block</td>
<td>251</td>
</tr>
<tr>
<td>Go To Block</td>
<td>255</td>
</tr>
<tr>
<td>Return Block</td>
<td>257</td>
</tr>
<tr>
<td>Segmentation Block</td>
<td>259</td>
</tr>
<tr>
<td>Shared Module Block</td>
<td>263</td>
</tr>
<tr>
<td>Terminate Call Block</td>
<td>267</td>
</tr>
<tr>
<td>Terminate Block (Digital)</td>
<td>269</td>
</tr>
<tr>
<td>Change Persona block</td>
<td>271</td>
</tr>
<tr>
<td><strong>User Interaction Blocks</strong></td>
<td></td>
</tr>
<tr>
<td>User Interaction Blocks</td>
<td>274</td>
</tr>
<tr>
<td>Bot Block</td>
<td>277</td>
</tr>
<tr>
<td>Menu Block</td>
<td>297</td>
</tr>
<tr>
<td>Menu Option Block</td>
<td>303</td>
</tr>
<tr>
<td>Play Message Block</td>
<td>309</td>
</tr>
<tr>
<td>Record Block</td>
<td>315</td>
</tr>
<tr>
<td>Record Utterance Block</td>
<td>322</td>
</tr>
<tr>
<td>User Input Block</td>
<td>326</td>
</tr>
<tr>
<td>Automated Message Block</td>
<td>336</td>
</tr>
<tr>
<td>Chat Message Block</td>
<td>341</td>
</tr>
<tr>
<td>Chat Custom Message Block</td>
<td>344</td>
</tr>
<tr>
<td>Send Chat Transcript Block</td>
<td>347</td>
</tr>
<tr>
<td>Get Chat Transcript Block</td>
<td>351</td>
</tr>
<tr>
<td>Send Email Block</td>
<td>353</td>
</tr>
<tr>
<td>Send SMS block</td>
<td>357</td>
</tr>
<tr>
<td><strong>Business Controls Blocks</strong></td>
<td></td>
</tr>
<tr>
<td>Business Controls Blocks</td>
<td>361</td>
</tr>
<tr>
<td>Business Hours Block</td>
<td>364</td>
</tr>
<tr>
<td>Block Type</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Data Tables Block</strong></td>
<td>374</td>
</tr>
<tr>
<td><strong>Emergency Block</strong></td>
<td>378</td>
</tr>
<tr>
<td><strong>Special Day Block</strong></td>
<td>382</td>
</tr>
<tr>
<td><strong>Routing Blocks</strong></td>
<td>386</td>
</tr>
<tr>
<td>Routing Blocks</td>
<td></td>
</tr>
<tr>
<td>Query VQs Block</td>
<td>389</td>
</tr>
<tr>
<td>Route Agent Block</td>
<td>392</td>
</tr>
<tr>
<td>Route Call Block</td>
<td>401</td>
</tr>
<tr>
<td>Routing Algorithms</td>
<td>416</td>
</tr>
<tr>
<td>Route Digital Block</td>
<td>420</td>
</tr>
<tr>
<td>Start Treatment Block</td>
<td>432</td>
</tr>
<tr>
<td>Transfer Block</td>
<td>436</td>
</tr>
<tr>
<td>Voice Mail Block</td>
<td>438</td>
</tr>
<tr>
<td>Predictive Routing Block</td>
<td>440</td>
</tr>
<tr>
<td><strong>Data Blocks</strong></td>
<td>445</td>
</tr>
<tr>
<td>Data Blocks</td>
<td></td>
</tr>
<tr>
<td>Call Data Block</td>
<td>448</td>
</tr>
<tr>
<td>Statistic Block</td>
<td>460</td>
</tr>
<tr>
<td>Transaction List Block</td>
<td>468</td>
</tr>
<tr>
<td><strong>External Services Blocks</strong></td>
<td>471</td>
</tr>
<tr>
<td>External Services Blocks</td>
<td></td>
</tr>
<tr>
<td>Custom Service Block</td>
<td>473</td>
</tr>
<tr>
<td>HTTP REST Block</td>
<td>478</td>
</tr>
<tr>
<td><strong>Reporting Blocks</strong></td>
<td>486</td>
</tr>
<tr>
<td>Reporting Blocks</td>
<td></td>
</tr>
<tr>
<td>Activity Block</td>
<td>488</td>
</tr>
<tr>
<td>Milestone Block</td>
<td>491</td>
</tr>
<tr>
<td>Debug Block</td>
<td>495</td>
</tr>
<tr>
<td><strong>Callback Blocks</strong></td>
<td>498</td>
</tr>
<tr>
<td>Callback Blocks</td>
<td></td>
</tr>
<tr>
<td>Callback V2</td>
<td>501</td>
</tr>
<tr>
<td>Book ASAP Callback V2</td>
<td>520</td>
</tr>
<tr>
<td>Book Scheduled Callback V2</td>
<td>523</td>
</tr>
<tr>
<td>Callback Availability V2</td>
<td>527</td>
</tr>
<tr>
<td>Cancel Callback V2</td>
<td>530</td>
</tr>
<tr>
<td>Check for Existing Callback V2</td>
<td>533</td>
</tr>
<tr>
<td>Validate Phone Number Block</td>
<td>536</td>
</tr>
<tr>
<td>Callback VQ Watermark Block</td>
<td>540</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td><strong>Survey Blocks</strong></td>
<td></td>
</tr>
<tr>
<td>Survey Blocks</td>
<td>543</td>
</tr>
<tr>
<td>Setup Survey Block</td>
<td>545</td>
</tr>
<tr>
<td><strong>Designer Analytics</strong></td>
<td></td>
</tr>
<tr>
<td>Designer Analytics</td>
<td>558</td>
</tr>
<tr>
<td>Dashboard management</td>
<td>562</td>
</tr>
<tr>
<td>Summary</td>
<td>569</td>
</tr>
<tr>
<td>Application Details</td>
<td>574</td>
</tr>
<tr>
<td>Durations</td>
<td>579</td>
</tr>
<tr>
<td>Data Tables Dashboard</td>
<td>582</td>
</tr>
<tr>
<td>Spikes</td>
<td>585</td>
</tr>
<tr>
<td>Heatmap</td>
<td>587</td>
</tr>
<tr>
<td>Path</td>
<td>590</td>
</tr>
<tr>
<td>Sankey Path Analysis</td>
<td>594</td>
</tr>
<tr>
<td>Sunburst Path Analysis</td>
<td>597</td>
</tr>
<tr>
<td>Inputs</td>
<td>600</td>
</tr>
<tr>
<td>Surveys</td>
<td>603</td>
</tr>
<tr>
<td>External Services</td>
<td>606</td>
</tr>
<tr>
<td>Routing Analysis</td>
<td>610</td>
</tr>
<tr>
<td>Business Control Dashboard</td>
<td>614</td>
</tr>
<tr>
<td>Session Detail Records</td>
<td>619</td>
</tr>
<tr>
<td>Bots Dashboard</td>
<td>623</td>
</tr>
<tr>
<td>Session Detail Record (SDR) Fields Reference</td>
<td>635</td>
</tr>
<tr>
<td><strong>ECMAScript in Designer</strong></td>
<td></td>
</tr>
<tr>
<td>ECMAScript Expressions</td>
<td>646</td>
</tr>
<tr>
<td><strong>Troubleshooting</strong></td>
<td></td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>653</td>
</tr>
<tr>
<td><strong>Store</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Designer Sample Applications</strong></td>
<td></td>
</tr>
<tr>
<td>Genesys Multicloud CX Email</td>
<td>664</td>
</tr>
<tr>
<td>SMART Inbound Voice Routing Application</td>
<td>672</td>
</tr>
<tr>
<td>Default Routing in Designer</td>
<td>680</td>
</tr>
</tbody>
</table>
Search the table of all articles in this guide, listed in alphabetical order, to find the article you need.

**Related documentation:**

•
Getting Started with Designer

Contents

• 1 Before you start
  • 1.1 Important!
  • 1.2 Supported browsers
  • 1.3 Minimum display resolution
  • 1.4 Third-party cookies

• 2 Access the application

• 3 Walk through the application
  • 3.1 Navigation bar
  • 3.2 Workspace toolbar
  • 3.3 Application toolbar
  • 3.4 Palette
  • 3.5 Actions toolbar
  • 3.6 Application flow
  • 3.7 Help pane
  • 3.8 Block properties
  • 3.9 Quick filters

• 4 Next steps

• 5 More about Designer
Designer is a web-based tool for developing self-service (IVR) and assisted service (routing) applications that run on the Genesys Multicloud CX platform. It is an omnichannel solution, enabling you to craft applications that handle voice, chat, and email interactions.

This video shows a quick overview of Designer and demonstrates a few of its latest features:

[Link to video]

Designer provides easy-to-use, highly functional blocks that enable common tasks in a simple and straightforward manner. For example, there are blocks for building menus, playing messages, setting up decision points, routing interactions to agents, and offering surveys.

To build an application, you simply drag blocks from the Palette and drop them into the Application Flow, which is divided into phases and represents the application structure.

For example, you might use the following blocks in the Assisted Service phase, in which an agent helps a customer. The first block controls the announcement to the caller that they are being transferred, and the second block controls the routing function to an agent:

Before you start

Important!

Some of the features and functionality described in these topics might not be applicable to all deployments. Contact your Genesys representative if you have any questions about what is available in your version of Designer.

Supported browsers

Unless otherwise noted, Designer supports the latest versions of the following browsers:

- Mozilla Firefox
- Google Chrome (see Important, below)
- Microsoft Edge
- Apple Safari

**Not supported:** Internet Explorer (all versions)
Important
For Google Chrome, Designer supports the n-1 version of the browser, i.e. the version prior to the latest release.

Minimum display resolution
The \textbf{minimum} display resolution supported by Designer is \textbf{1920 x 1080}.

Third-party cookies
Certain features in Designer require the use of third-party cookies. Browsers must allow third-party cookies to be stored for Designer to work properly.
Access the application

Once your Genesys Multicloud CX environment is up and running and you've checked that you meet the necessary requirements, log in to your Genesys Portal to access Designer. Click the Designer icon and enter your username and password.

Walk through the application

Watch this video to see an overview of the Designer user interface:
The various elements within the interface are described below.

Navigation bar

Provides one-click access to Applications, Shared Modules, Media Resources, Digital 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. Click Build to create and manage the builds associated with the application.
Palette

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

Actions toolbar

Many of the resource pages in Designer have an **Actions** toolbar. The action items on the toolbar will vary based on the resource page you are viewing. For more information, see Actions toolbar.
Application flow

Provides the main area to build your application by adding blocks. For more information, see Using the blocks.

Help pane

Displays help information for the selected block.
### Block properties

**Menu Prompts**  
Specify prompts to play to offer menu selection  

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td>□</td>
<td>Press 1 for sales.</td>
<td>text</td>
<td>▼ ▲</td>
</tr>
<tr>
<td>TTS</td>
<td>□</td>
<td>2 for service.</td>
<td>text</td>
<td>▼ ▲</td>
</tr>
<tr>
<td>TTS</td>
<td>□</td>
<td>3 to check if there are any suppl...</td>
<td>text</td>
<td>▼ ▲</td>
</tr>
</tbody>
</table>

Timeout - wait for **5** s before assuming that no input was received.

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. For more information, see Using the Quick Filters.

### Next steps

Use this page as a starting point to learn more about key Designer features and concepts:

- Quick links
More about Designer

Link to video

This video highlights several of the new features and capabilities that were added to Designer during this past year (2021). It also shares a few customer success stories and offers a quick preview of what's ahead for Designer.

Important

The information provided in this video is for general informational purposes only. Some features or functionality described may not apply to your deployment or may have changed since the date of original production. If you have any questions, contact your Genesys representative.
Quick links

• Administrator

Learn about common tasks for administrators in Designer.

Related documentation:

Contents

• 1 Getting started
• 2 Managing resources
• 3 Building applications
• 4 Reporting and Analytics
Use these topics as a starting point to find the information you need.

Getting started
Learn the basics.

• Getting started with Designer
• Using the Designer user-interface
• Understanding the application workflow
• Designer roles and permissions
• Managing variables

Managing resources
View and manage your application resources.

• Business Controls (Business Hours, Special Days, Emergency Flags, Data Tables)
• Media resources and collections
• Digital messaging resources
• Speech grammars
• Bot Registry

Building applications
Using the various blocks to build applications in Designer.

• Application phases
• Working with blocks

Reporting and Analytics
Designer has an integrated set of dashboards that provide detailed information about your operations.
Quick links

- Designer Analytics
- Managing the dashboards
Permissions and Access

Contents

• 1 Designer roles
  • 1.1 Designer Developer
  • 1.2 Designer Business User
  • 1.3 Designer Administrator
  • 1.4 Designer Analytics

• 2 Designer permissions
  • 2.1 Media Resources
  • 2.2 Message Resources
  • 2.3 Applications
  • 2.4 Shared Modules
  • 2.5 Speech Grammars
  • 2.6 Bot Registry
  • 2.7 Business Controls

• 3 Restricted mode during upgrades
Learn about the various roles and permissions in Designer.

**Related documentation:**

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 can 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 Developers can also access PII Rules Management and Standard Responses, if these management tools are available in their environment.

**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 control the resources that users have access to through **Partition-Based Access Control**, or **PBAC**.

With PBAC, you can create a partition and assign certain Designer resources to it. Then you can select the users who will belong to each partition. Users will only be able to see and manage those resources that are assigned to the partitions they belong to. For more information, see Partitions.

Designer Administrators 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

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Message Resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Permissions and Access

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch to last snapshot</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Assign phone numbers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Change block properties</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Shared Modules

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Speech Grammars

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bot Registry

<table>
<thead>
<tr>
<th>Task</th>
<th>Designer Developer</th>
<th>Designer Business User</th>
<th>Designer Administrator</th>
<th>Designer Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Modify</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.
Video Gallery for Designer

All Designer videos, collected on one page.

**Related documentation:**

Selected Videos

**Using bots in Designer applications**
From:

**Adding a bot resource to the registry**
From:
Building a chat-based survey

From:

Designer overview

From:

Designer search tools

From:
How Partition-Based Access Control works
From:

Managing Personally Identifiable Information (PII) in Designer
From:

Setting up Partition-Based Access Control
From:
The Designer user-interface
From:

Using cascaded routing to prioritize routing
From:

Viewing the history of applications and resources
From:
Working with application builds and streams
From:

Contents

- 1 Selected Videos
Upgrade notes

Contents

• 1 Changes that take effect immediately
• 2 Changes that require actions to be taken
• 3 Designer and other Genesys services
• 4 If you've migrated to Designer 9 from Designer 8.x
  • 4.1 New workflow
  • 4.2 Changes to resources
  • 4.3 Changes to roles and permissions
• 5 Getting help
Important information about Designer upgrades.

**Related documentation:**

- Administrator

When Designer is upgraded to a new version, certain changes take effect immediately while others require additional actions to be taken before they take effect.

Designer Release Notes may also contain information specific to new features and changes, such as exceptions, recommendations, or additional considerations. In those cases, you should follow the guidance provided in the Release Notes. If you have any questions about how a Designer upgrade will impact your operations, check with your Genesys representative.

**Changes that take effect immediately**

Certain Designer blocks use runtime APIs that are automatically updated whenever Designer is upgraded to a new version. There is no ability to opt out or defer these types of changes. Therefore, any changes to the following Designer blocks will take immediate effect after an upgrade, with no further actions required:

- Business Controls blocks (Business Hours, Special Days, Emergency Flags, Data Tables)
- HTTP REST block

Changes introduced in other blocks by the new version of Designer do not take effect until the application that uses these blocks is republished.

**Changes that require actions to be taken**

Additional actions are almost always required for existing applications that are known to be affected by an issue that is resolved in the new version of Designer. In some cases, a resolved issue may be minor enough to only require that you generate a new build to incorporate the changes. However, to ensure your applications continue to work as intended after an upgrade, it is highly recommended that you perform the following steps:

- Publish the application and generate a new build. The new build picks up the modified behavior in the new version of Designer.
- Assign the new build to the DEV, QA, or UAT streams and test to confirm that the new feature or resolved issue is implemented and working correctly. (This follows the standard Designer application workflow.)
- If the testing is successful, assign the build to the LIVE or LIVE_B stream. You can then observe the build for a period of time in Designer Analytics to confirm that it is working correctly.
If you experience an issue with the new build, you can perform a rollback by assigning the build that was working previously to the **LIVE** or **LIVE_B** stream. This enables you to continue troubleshooting the problem without impacting your production traffic.

Designer and other Genesys services

Designer includes functionality that is provided or supported by other Genesys cloud services. The release notes for the following services may also contain items that pertain to Designer:

- Digital Channels

If you've migrated to Designer 9 from Designer 8.x

If you are new to Designer 9 after working in a previous version (i.e. Designer 8.x), there are some changes you should be aware of when working in Designer 9:

New workflow

- Designer now uses a new application workflow based on builds and streams. You can learn more about this new type of workflow on the Application workflow page. To gain familiarity, you can clone an existing application and experiment with making changes, generating builds, and assigning the builds to streams. When you are comfortable with the workflow, you can start to adopt it with your original applications.

Changes to resources

- If you make any changes to a **Business Control**, **Media Resource**, or **Speech Grammar** resource, those changes now take effect immediately if the resource is being referenced by the LIVE production build. However, if you make a change to a **Data Table**, you must publish the data table for the changes to take effect in any builds that are referencing it.

- In previous versions of Designer, any changes made to an application took effect immediately after the application was published. Now, you must generate a new build and assign it to the LIVE stream for the changes to take effect.

Changes to roles and permissions

- Some permissions have changed. Refer to Permissions and Access for a list of Designer roles and their associated permissions.

Getting help

If you need additional help after your Designer upgrade, contact your Genesys representative or visit the Genesys Customer Care Communications portal.
Applications

Contents

• 1 Application types
  • 1.1 Default
  • 1.2 Digital
  • 1.3 Callback
• 2 Creating a new application
• 3 Viewing application properties
• 4 Edit the application name, description, or tags
• 5 Change the status of an application stream
• 6 Viewing application history
• 7 Clone an application
• 8 Manage Builds
• 9 Manage Phone Numbers
  • 9.1 Batch Assign
  • 9.2 Transfer from Others
• 10 Manage Mail Boxes
• 11 Manage Open Media endpoints
• 12 Manage Chat Endpoints
• 13 Manage Digital Endpoints
• 14 Disconnect Phone Numbers and Chat Endpoint
• 15 Enabling Your Application
• 16 Viewing the application summary
  • 16.1 Filtering
  • 16.2 Session playback
• 17 Application settings
Applications

• Administrator

Create new applications and make changes to existing ones.

**Related documentation:**

From the **Applications** page, you can create new applications and manage existing ones.

Application types

Designer has three primary application types:

**Default**

These applications support Interactive Voice Response (IVR) and voice call routing. They also support chat if the omnichannel option is enabled in the application settings.

**Important**

When a Default type application is enabled for omnichannel, it can handle voice, chat, and SMS interactions. However, note that the **Menu** block does not support SMS media.

**Digital**

This application type supports digital interactions only, such as email and open media (no voice calls).

**Callback**

This application type is used for setting up outbound callbacks (voice calls only). Digital interactions are not supported. For more information about using Callback with Designer, see Provisioning Callback in Designer in the **Callback Administrator's Guide**.
Creating a new application

To create a new application, click Add Application.

Enter a Name for the application and select the Type of application you want to create.

Click Create to save it and go back to the Applications page. Or, click Create and Open to save the application and open it for editing. To learn more about how to build applications using the blocks in Designer, see Using the blocks.

Viewing application properties

From the Applications list, click an empty spot in an application's row to display its properties on the right-hand side (if you click the application name, it will open for viewing or editing):
Edit the application name, description, or tags

Click the pencil icon to open the editing view.

From the editing view, you can change the application name, description, or manage the tags associated with the application.
Tip
You can also view or make changes to the application name and description by clicking the application name when its flow is open for editing.

Change the status of an application stream

Use the sliders to change the status of an application stream to enabled or disabled.

To enable:

To disable:

Viewing application history

Click Application History to see a list of historical events for the application, such as each time it was viewed, edited, or published. You can then drill-down into each event to review its audit details. For more information, see History View.
Clone an application

Click **Clone application** to create a new application that is an exact copy of the selected application. Save the clone with a unique name and then manage it like any other application. This can be useful when you want to use the same structure as the selected application, but need to specify different settings or resources.

Manage Builds

Click **Manage Builds** to manage the application builds that are available for the application.

All builds are initially enabled by default. To disable a build, deselect it.

To view the application flow for a build, click the **View Build** icon under the **Actions** column. This displays the application flow as it existed when the build was generated (in read-only mode).

To delete a build, you must first disable it. This reveals a trash icon under the **Actions** column that you can click to delete the build.

For more information about builds, see Application builds.

Manage Phone Numbers
Click **Manage** in the **Phone Number** column to select and assign a phone number(s) to the application.

When assigning a phone number:

- 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 can indicate:
  - The phone number was not assigned in 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.

**Batch Assign**

What if you have a large list of phone numbers that you want to assign to the application? With **Batch Assign**, you can copy and paste the phone numbers into a form and assign them in a single operation.

Here’s how it works:

Click **Batch Assign** and add the numbers you want to assign to the input column. Note that the numbers must be on separate lines. Designer keeps track of the numbers you’ve added in the **Recognized phone numbers** column.

When you are finished adding numbers, click **Next (review)**.

You can review the results to see the phone numbers that are ready to be assigned, are already assigned to this application (or another application), or were not found — maybe these were entered incorrectly or are not available for use.

If you need to make changes, click **Previous**. Otherwise, click **Assign the numbers**.
You can monitor the progress as Designer assigns the phone numbers to the application:

When the operation completes, Designer shows you which numbers were successfully assigned and which (if any) failed.
Click **OK** to close the window.

**Transfer from Others**

Use **Transfer from Others** to transfer multiple phone numbers to another application or to other stream of the current application. To show an example of how this works, let's transfer a phone number to another stream of an application.

We've clicked **Manage Phone Numbers** for the Joules Coulomb application and selected **Transfer from Others**:
We want to transfer a number to a stream of the current application, so we use the search bar to filter the application list for "Joules". We select it, and the stream tabs appear at the bottom.

We want to move a number from LIVE to DEV, so we select the LIVE tab.

There are two numbers assigned to that stream, but we only want to move one of them. We select the number we want to move and click Next (review).

Review the transfer details:
If you need to make any changes, click Previous. If everything looks ok, click Confirm.

Designer shows you the progress of the transfer. When it is complete, the summary lists the numbers that were successfully transferred and also any that failed.

Click OK to close the transfer results window and return to the application properties.

Manage Mail Boxes

Click Manage in the Mail Boxes column to select and assign a mail box to the application.
• This option is available for **Digital** type applications only.
• The mail box is assigned only to the application stream for which it was selected.
• Mail Boxes are managed and configured in Agent Setup. For more information, contact your Genesys representative.

Manage Open Media endpoints

![Open Media endpoints table]

Click **Manage** under **Open Media** to select and assign an Open Media endpoint to the application.

When assigning an Open Media endpoint:

• This option is available for **Digital** type applications only.
• The endpoint is assigned only to the application stream for which it was selected.
• Open Media endpoints are managed and configured in **Agent Setup**. For more information, contact your Genesys representative.

Manage Chat Endpoints
Click **Manage Chat Endpoints** column to assign a chat endpoint to the application.

When assigning a chat endpoint:

- You can only assign digital endpoints in **Default** type applications that are enabled for **omni-channel support** (see the digital tab in Application Settings).
- 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 ok).
- The selected endpoint is assigned to each of the application streams, with a naming extension that corresponds to the stream. For example, *ChatEndpoint>_dev, _qa*, and so on.

Manage Digital Endpoints
Click **Manage** in the **Digital Endpoints** column to select and assign a digital endpoint to an application stream. Digital endpoints include platforms such as Facebook, Twitter, SMS, and WhatsApp.

When assigning a digital endpoint:

- You can only assign digital endpoints in **Default** type applications that are enabled for **omni-channel support** (see the digital tab in Application Settings).
- The endpoint is assigned only to the application stream for which it was selected.
- Digital endpoints are managed and configured outside of Designer. For more information, contact your Genesys representative.

**Disconnect Phone Numbers and Chat Endpoint**

Click **Disconnect Phone Numbers and Chat Endpoint** to disconnect all phone numbers and the chat endpoint from the application (you might need to click **More** to see this option). Once disconnected, the phone numbers or endpoint can be assigned to other applications.

If you want to disconnect specific numbers from an application stream, click **Manage** under the **Phone Numbers** column for that stream to deselect the number(s) you want to disconnect.

**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:

![Diagram of a Segmentation node showing decision points for routing calls.]

**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:

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

**Application settings**
To view and manage the settings for an application, see Application settings.
Application Settings

Contents

• 1 Managing the application settings
• 2 General
• 3 Media
  • 3.1 Media Resource Collection
  • 3.2 External Resources
• 4 Persona
• 5 Messaging
• 6 Reporting
• 7 DTMF Options
• 8 Speech Recognition
• 9 Global Retry
  • 9.1 Number of No Input retries allowed
  • 9.2 Number of No Match retries allowed
  • 9.3 After Final No Input
  • 9.4 After Final No Match
• 10 Digital
  • 10.1 Enable omni-channel support for Default applications
  • 10.2 Designer Application party name
  • 10.3 Continue processing when the customer disconnects
  • 10.4 Terminate the interaction when the customer disconnects
  • 10.5 Terminate the interaction when the interaction is stale
• 11 Predictive Routing
• 12 Contextual Data
  • 12.1 Interaction history data
• 13 Caching
• 14 Misc
  • 14.1 PTE (Parallel Test Environment)
  • 14.2 Tracing
Application Settings

- 14.3 Wait Answering Machine
- 15 Language
Managing the application settings

When an application is open for editing, you can click Settings to view and manage the settings for that application.

General

- **Application Reporting Title** - Specify the name to use for the application 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.

Media
Use this tab to specify the media settings for the application.

Media Resource Collection

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

External Resources

**Fetch Timeout**—If you want to override the default fetch timeout for external audio resources (500ms), enter a new value (in milliseconds) here.

This option is applicable to **Default** application types only.

Persona
Use this tab to manage the Personas settings for the application.

- Select **Enable Persona** to enable the use of personas in the application.
- Select **Enable Custom Voices** if you are using a Google Text-to-Speech (TTS) custom voice.

If the application uses a Google TTS voice (regular or custom) for a persona and invokes a Google Dialogflow CX voice bot, the same voice is used by the bot when rendering next turn audio.

For more information, see the Personas page.

**Messaging**

- This setting is only available for **Digital** type applications.
- **Digital Resource Collection** - Select the Digital Resource collection that this application will use. See the Digital Resources page for more information.

**Reporting**

- **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**

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

**Global Retry**
This tab lets you specify a global retry prompt to play if a customer provides input that is not recognized. When **Allow Retries** is enabled, you can specify the **No Input** and **No Match** settings. These settings are described in more detail in the section below.

### Number of No Input retries allowed

Select the number of retries to allow for customers who do not provide any 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 customers who 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.

Digital
The **Digital** tab of the application settings allows you to enable omni-channel support for Default applications. When enabled, it allows your application to handle both voice and chat interactions.

You can also specify additional options for what Designer should do in certain scenarios, such as when the customer disconnects from the session.

**These options are described in more detail in the section below.**

**Enable omni-channel support for Default applications**

If your application is a Default type, enabling this option allows the application to handle both voice and chat interactions. For example, during chat sessions, the application displays voice prompts and menu options as chat messages.

Once enabled, the following additional options are available:

**Designer Application party name**

You can (optionally) enter a name for the application to use as the chat participant. This is the name that customers will see displayed in the chat widget when they join the chat session. If you don't specify a party name, Designer uses the **Application Reporting Title** as specified in the General settings.

---

The **Digital** tab of the application settings allows you to enable omni-channel support for Default applications. When enabled, it allows your application to handle both voice and chat interactions.

You can also specify additional options for what Designer should do in certain scenarios, such as when the customer disconnects from the session.

**These options are described in more detail in the section below.**

**Enable omni-channel support for Default applications**

If your application is a Default type, enabling this option allows the application to handle both voice and chat interactions. For example, during chat sessions, the application displays voice prompts and menu options as chat messages.

Once enabled, the following additional options are available:

**Designer Application party name**

You can (optionally) enter a name for the application to use as the chat participant. This is the name that customers will see displayed in the chat widget when they join the chat session. If you don't specify a party name, Designer uses the **Application Reporting Title** as specified in the General settings.
Continue processing when the customer disconnects

This option is enabled by default for digital interactions but does not apply to voice interactions. Keep this option enabled if you want Designer to continue processing the application after the customer leaves the chat session. If the customer disconnects, Designer moves on to the next block and continues processing blocks that do not require any interaction or involvement with the customer, such as executing an HTTP request or scripting logic, but skips any blocks that would require the customer to be present (such as sending a message or requesting input).

If you disable this option, Designer immediately terminates the chat session when the customer leaves and jumps directly to the Finalize phase. Note that this can interrupt any on-going processing, including routing.

Terminate the interaction when the customer disconnects

This option is also enabled by default and terminates the interaction when the customer leaves the chat session. If the Continue processing when the customer disconnects option is not enabled, Designer jumps immediately to the Finalize phase when the customer disconnects from the session.

However, if both this option and the Continue processing when the customer disconnects option are enabled, Designer continues processing blocks that do not require any interaction or involvement with the customer, such as executing an HTTP request, while skipping any blocks that would require the customer to be present (such as sending a message or requesting input).

Note that this can create a delay in closing the interaction that can cause Designer to sometimes report the interaction as Abandoned. If you want Designer to terminate the interaction immediately when the customer disconnects, enable this option but make sure that Continue processing when the customer disconnects is disabled.

Terminate the interaction when the interaction is stale

This option is enabled by default and terminates the chat interaction if it has gone stale.

Important

If your application is using the same TTS prompts and messages for both voice and chat channels, any SSML tags being used will be displayed in chats. To prevent this, set the removeSSMLInChat system variable to true.

Predictive Routing
Select this option to enable support for predictive routing (as required for the Predictive Routing block).

Contextual Data

The Enable Customer Profile option tells Designer to retrieve certain details about the customer associated with the incoming interaction (such as their name, phone number, and email address). For voice calls, Designer identifies the caller based on their phone number (ANI). For digital interactions, the contact’s email address is used. You can then choose how much interaction history data to query for each customer.

This contextual data can then be used in the application to provide a more personalized experience for the customer—for example, by playing announcements that are specifically designed for them or using the Last Called Agent routing option to direct the customer to the last agent they spoke with.

Designer stores contact details in the Contact system variable. For more information, see system variables.
Important
This option must be enabled if you are using the Last Called Agent option in the Route agent block.

Interaction history data

When the Enable Customer Profile option is enabled, you can choose how much interaction history data to retrieve for the customer. By default, Designer retrieves 30-days of interaction history data, but you can adjust this to 1 hour, 24 hours, or choose to disable the interaction history entirely.

Designer provides the following functions to work with the interaction history data:

- `getInteractionCount()` — returns the number of interactions for the contact.
- `getLastInteractionDetails()` — returns a JSON object that contains details about the last interaction with the contact.

To use the results in an application, you can create user variables in which to store the results for each function and then use an Assign Variables block to assign the function expression to the appropriate variable. For example:

Caching

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**

From the **Misc** tab, you can enable the **Parallel Test Environment**, **Tracing**, and **Wait Answering Machine** options.

**PTE (Parallel Test Environment)**

Enable the **PTE** (Parallel Test Environment) option if you want non-production streams (DEV, QA, UAT).
to use test versions of resources instead of the resources being used in the LIVE production environment.

For more information, see Parallel Test Environment (PTE) on the Application Workflow page.

**Tracing**

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.

**Wait Answering Machine**

This option tells Designer to wait for an answering machine beep (i.e. the tone that signals the caller to begin recording their message) during an outbound call. For example, if the customer does not answer the call, the application waits for an answering machine beep before playing a message.

**Language**

![Language Settings](image)
From the **Language** tab, you can select the languages to be used for playing intelligent prompts.

You can select multiple languages, up to a maximum of eight. When playing intelligent prompts, Designer uses the overall language setting in the application (for example, the language specified by the **Language** system variable or a Change Language block) to determine which language to use.

**Tip**

Genesys recommends reviewing these selections regularly (for example, when publishing the application) and deselecting any languages that are not needed.
Application phases

Contents

• 1 Initialize
• 2 Self Service
• 3 Assisted Service
• 4 Finalize
• 5 Dispositions
  • 5.1 General
  • 5.2 Error
  • 5.3 Self Service
  • 5.4 Assisted Service
  • 5.5 Callback
Application phases

Application flows are comprised of sections of common blocks, known as application phases. Learn about the role each phase plays within the Designer application flow.

**Related documentation:**

Each application flow in Designer contains four phases:

- **Initialize**
- **Self Service**
- **Assisted Service**
- **Finalize**

You can learn more about each phase in the sections below.

**Initialize**

This phase initializes application-level user variables and parameters to use when the application executes. The application initializes during this phase.

By default, the following actions take place:

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

**Self Service**
The **Self Service** phase is the IVR portion of the interaction. This phase hosts blocks that provide automated interaction with the customer via speech, chat, and/or DTMF. It attempts to provide automated service and contain the interaction within an IVR, so there is no need to route the interaction 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 interaction in the next phase, **Assisted Service**.

---

**Tip**

To enable voice call recording for the Self Service phase, set the `EnableSSRecording` variable to `true` in the **System Variables** section.

---

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

- Play Messages. These may be pre-recorded audio files or dynamic text spoken using TTS.
- Check business hours and customize logic based on the outcome (for example, take *this* action if we are closed).
- Collect user input.
- Present choices to customers using menus.
- Navigate customers appropriately, based on their responses (segmentation and branches).
- Call external RESTful APIs and fetch data into user variables.
- Update user variables and write ECMAScript expressions.
- 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.

Interaction 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**

This phase hosts blocks that route the interaction to a live agent, if necessary.

During the **Assisted Service** phase, the application attempts to route interactions 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:
• Attempt to route the call while playing music or prompts.
• Call external RESTful APIs.
• Update user variables.

There may be multiple Route Call blocks in sequence. Each Route Call block might try to route the interaction 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

This phase provides post-processing and interaction termination after the customer has been serviced. When interaction processing is finished, the application goes to the Finalize phase to perform post-processing for various scenarios that are based on how the interaction was completed.

The following are examples of typical scenarios:

• Interaction was abandoned by the customer (while in either the Self Service or Assisted Service phase).
• Interaction was completed in Self Service phase.
• Interaction was routed to an agent in the Assisted Service phase.
• Interaction was delivered to voicemail in the Assisted Service phase.
• 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 Application phases for post-processing. When you click on the Finalize block in the application flow, each of the tabs has a list of dispositions that you can select. 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:
Tip

Setting up handlers for the **Finalize** phase is optional. You might not need to do anything special for these cases.

---

Dispositions

This section describes the dispositions that can be selected in the Finalize phase.

General

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All</strong></td>
<td>When an application enters the <strong>Finalize</strong> phase, it has only one disposition code, so only one disposition block is executed. However, the <strong>All</strong> 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 <strong>All</strong> 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.</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>This code is used when no other disposition code is applicable. For example, it could indicate that an interaction was not routed, not terminated (by any</td>
</tr>
</tbody>
</table>
### Disposition

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revoked</td>
<td>The Designer application can no longer control the interaction due to external reasons unrelated to the application logic or processing.</td>
</tr>
<tr>
<td>Terminated (Terminate Call)</td>
<td>The application reached a Terminate Call Block.</td>
</tr>
<tr>
<td>Terminated (Business Hours)</td>
<td>The interaction arrived outside of regular business hours, as specified in the Business Hours Block.</td>
</tr>
<tr>
<td>Terminated (Special Days)</td>
<td>The interaction arrived on a special day, as specified in the Special Day Block.</td>
</tr>
<tr>
<td>Terminated (Auto-Stop)</td>
<td>The interaction was processed multiple times by the Designer application and exceeded the defined threshold, or a single application session lasted beyond the MaxTime defined in the application (see System Variables). The session and interaction were auto-terminated.</td>
</tr>
<tr>
<td>Default Routed</td>
<td>The interaction was delivered to the default routing destination.</td>
</tr>
<tr>
<td>Interaction Less</td>
<td>Used when a consult interaction is merged with a parent interaction that the application session is not able to take ownership of. Without an interaction to associate with the session, the application applies this disposition code and proceeds to the <strong>Finalize</strong> phase.</td>
</tr>
</tbody>
</table>

### Error

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Error</td>
<td>There was an unexpected error in the application (such as a script validation error).</td>
</tr>
</tbody>
</table>

### Self Service

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandoned in Self Service</td>
<td>The customer hung up before completing the <strong>Self Service</strong> phase of the application.</td>
</tr>
<tr>
<td>Completed in Self Service</td>
<td>The customer successfully completed their interaction in the <strong>Self Service</strong> phase — the application did not need to go to <strong>Assisted Service</strong> and went directly to <strong>Finalize</strong>. If the interaction ended in <strong>Self Service</strong> due to some other condition or event (such as the interaction being received outside of business hours or the customer hanging up), then the appropriate disposition code for that condition or event is used (<strong>Abandoned in Self Service</strong>, <strong>Terminated</strong>, and so on).</td>
</tr>
<tr>
<td>Terminated (Emergency)</td>
<td>The emergency flag was set, as specified in the <strong>Application phases</strong>.</td>
</tr>
</tbody>
</table>
### Assisted Service

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandoned in Queue</td>
<td>The customer completed the <strong>Self Service</strong> phase, but was disconnected while waiting to speak with an agent.</td>
</tr>
<tr>
<td>Routed to Agent</td>
<td>The interaction was successfully delivered to an agent.</td>
</tr>
<tr>
<td>Routed to DN</td>
<td>The interaction was successfully delivered to a Direct Number.</td>
</tr>
<tr>
<td>Routing Incomplete</td>
<td>The interaction was not delivered to the target destination.</td>
</tr>
<tr>
<td>Routed to Voicemail</td>
<td>The interaction was delivered to voicemail.</td>
</tr>
<tr>
<td>Routed to Queue</td>
<td>The interaction was delivered to a virtual queue.</td>
</tr>
<tr>
<td>Routed to Parking Queue</td>
<td>The interaction was sent to a &quot;parking&quot; queue to wait until the business is open.</td>
</tr>
</tbody>
</table>

### Callback

The following final dispositions can be set by the Callback V2 block. However, this disposition value can be overwritten by any subsequent block that the Designer application enters.

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback Not Offered</td>
<td>Callback was not offered to the customer.</td>
</tr>
<tr>
<td>Callback Accepted</td>
<td>The customer accepted the callback offer.</td>
</tr>
<tr>
<td>Callback Declined</td>
<td>The customer declined the callback offer.</td>
</tr>
<tr>
<td>Callback Cancelled</td>
<td>The callback was cancelled.</td>
</tr>
<tr>
<td>Callback Rescheduled</td>
<td>The callback was rescheduled.</td>
</tr>
<tr>
<td>Callback Abandoned in Queue</td>
<td>The customer was reconnected, but abandoned the interaction queue while waiting for an agent.</td>
</tr>
<tr>
<td>Callback Routed to Agent</td>
<td>The customer was reconnected and routed to an agent.</td>
</tr>
<tr>
<td>Callback Outbound Failed</td>
<td>The customer was not successfully reconnected after the maximum number of attempts.</td>
</tr>
<tr>
<td>Callback Keep Existing</td>
<td>The customer had already booked a callback. When they called back in, they were informed they already had a callback. They chose to keep their existing callback and disconnected.</td>
</tr>
<tr>
<td>Callback Push Failed</td>
<td>The callback Push Notification failed to be sent to</td>
</tr>
</tbody>
</table>
## Application phases

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback Booking Failed</td>
<td>The callback failed to be booked.</td>
</tr>
<tr>
<td>Callback Purged</td>
<td>The callback was purged from the system.</td>
</tr>
<tr>
<td>Callback Matched</td>
<td>The customer called-in and was matched to an existing callback request.</td>
</tr>
<tr>
<td>Callback Push Expired</td>
<td>The Push Notification expired before the customer responded to it.</td>
</tr>
<tr>
<td>Callback Push Delivered</td>
<td>The Push Notification was successfully delivered to the customer's device.</td>
</tr>
<tr>
<td>Callback Call Moved</td>
<td>The callback was moved to another session.</td>
</tr>
</tbody>
</table>
About application resources

Contents

• 1 Types of resources
• 2 Actions toolbar
• 3 Using the Quick Filters
  • 3.1 Managing Quick Filters
• 4 Using the history view
• 5 Valid naming characters
• 6 File size limitations
  • 6.1 Uploads and imports
  • 6.2 Downloads and exports
  • 6.3 My file is too large – what can I do?
This page provides general details about working with resources in Designer.

**Related documentation:**

- Administrator

## Types of resources

From the Designer main navigation menu, you can access and manage a variety of resources that can be used in your applications, such as:

- Business Controls for setting up and manage your hours of business, holidays (and other special days), emergency flags, and data tables.
- Bot Registry for adding the bot resources you want to use in your applications.
- Speech Grammars for defining lists of phrases or options that a customer can provide as input to an application.
- Media Collections for managing voice recordings and other audio-related settings.
- Digital Resources for managing resources related to digital applications, such as standard responses and customized messages for chats.
- Shared Modules, which are smaller applications that you can use in other applications.
- Admin resources for managing resources available to users with Designer Administrator role privileges (if you are not assigned this role, you will not see this item in the navigation bar).

You can learn more about each resource on its respective page.

## Actions toolbar

Many of the resource pages in Designer have an **Actions** toolbar:

![Actions toolbar]

The action items on the toolbar will vary based on the resource page you are viewing. Depending on the functions available for that particular resource, you can do the following:
### About application resources

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Displays the history (or audit log) for a resource. For more information, see Resource history.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Displays a list of the applications and modules that are using a particular media or message collection.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Clones the resource. You'll need to assign the cloned resource a unique name. Note that a cloned resource does not inherit the history and published versions of the original resource.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Displays the version history for a shared module. You can then choose to make a version public, or view a version (in read-only mode) as it appeared at a particular time.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Exports the resource for use in another Designer workspace. When you export a resource, all versions of that resource are exported, including the unpublished version. If you are using a Safari browser when performing the export, the exported file is downloaded as unknown. The file is valid and can be imported successfully, but you can rename it to something more meaningful.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Imports the contents of a CSV file into a selected data table. For more information, see Import on the Data Tables page.</td>
</tr>
<tr>
<td><img src="#" alt="Button" /></td>
<td>Deletes all versions of the resource. However, note that published applications that were already using the resource (i.e. applications that had previously generated their code) are not affected.</td>
</tr>
</tbody>
</table>

### Using the Quick Filters

**Quick Filters** enable 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:
The filters 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.

You can use Quick Filters on the following resource pages:

- Special Days
- Business Hours
- Data Tables
- Applications
- Shared Modules
- Emergency Flags
- Media Resources
- Digital Resources.

The same filters appear on each resource page, and any filters that you create are visible to other Designer users.

Managing Quick Filters

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.

For example, to associate the SMART filter with the M1App tag:
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.

Using the history view

The history view enables you to see a list of events for a resource that provide details about each time it was accessed or changed. You can see details such as the ID of the user who initiated the event and the new and previous values of any resource properties that were changed.

Resources that provide this option include Applications, Shared Modules, Announcements, Grammars, and Business Controls.

Watch this short video to learn more about viewing the history of applications and resources:

Link to video

Depending on the resource you are viewing, there are a few different ways you can access the history
view. You can select the **History** button from the resource properties:

### Media Resource detail

**Name**

cb_already_placed

**Description**

Add description

**Tags**

Add a tag

[Delete]  [History]  [Reset]  [Save]

Or, if you are viewing a resources page, you can select a resource and click the **History** button on the **Actions** toolbar:

For applications, you can view the history by selecting that option from the application properties.

For example, here is the history view for an application:

You can use the buttons on the history page to filter the results for a specific time period (for example, **last 1W** to see the results for previous week), or use the date fields to specify a custom date range (you can go back up to a maximum of one year from the current date). Results can be sorted or searched.

To download the results, click **Export** to generate and save the results as a CSV file.

You can drill-down further into an historical event by double-clicking it. This displays the audit details...
for that particular event:

The **Values** tab displays information about the event, such as the resource type, the type of action taken, and the unique IDs of data relevant to the event.

If applicable, the **Diff** tab shows you information about the specific objects or properties that were impacted by that particular event. For some items, you can expand the item to display additional details:
Valid naming characters

When naming your resources, you can use the following characters (except where otherwise noted):

<table>
<thead>
<tr>
<th>Character or Symbol</th>
<th>Description</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-Z</td>
<td>uppercase alphabetical</td>
<td></td>
</tr>
<tr>
<td>a-z</td>
<td>lowercase alphabetical</td>
<td></td>
</tr>
<tr>
<td>0-9</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>period (or dot)</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>colon</td>
<td>Not valid for bot resources.</td>
</tr>
<tr>
<td>-</td>
<td>hyphen</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>underscore</td>
<td></td>
</tr>
<tr>
<td>@</td>
<td>&quot;at&quot; sign</td>
<td>Not valid for bot resources.</td>
</tr>
<tr>
<td>!</td>
<td>exclamation mark</td>
<td>Not valid for bot resources.</td>
</tr>
<tr>
<td>+</td>
<td>plus sign</td>
<td></td>
</tr>
</tbody>
</table>

Important
Resource names cannot be longer than 50 characters.

File size limitations

When managing resource files, note the following file size limitations. If your file exceeds these size limits, Designer is unable to complete the operation.

Uploads and imports

- Files for audio or speech grammar resources cannot exceed **10 MB**.
- Imported applications or shared modules cannot exceed **10 MB**.
- Imported audio collections cannot exceed **15 MB**.

Downloads and exports

- Applications or shared modules exported from Designer cannot exceed **150 MB (with audio)** or **50 MB (without audio)**.
- Exported audio collections cannot be larger than **100 MB**.
My file is too large – what can I do?

If your file exceeds the size limit, you can try the following options to decrease the file size:

- For importing zip files that contain multiple (bulk) audio files, split the file into multiple smaller zip files and upload each file separately.
- For audio or speech grammar files, try compressing the files before uploading.
- For DAR files that are too large to import, try reducing the file size by deleting large binaries or files that are not needed.
Using the blocks

Contents

• 1 Build logic
  • 1.1 Moving and arranging blocks
  • 1.2 Limiting Application Indentation
• 2 Search tools
• 3 ECMAScript Expressions
• 4 Busy Treatments
  • 4.1 Routing blocks
  • 4.2 Start Treatment block
  • 4.3 Example of a Shared Module treatment
• 5 Validation
Using the blocks

• Administrator

Learn how functional blocks are used to build Designer applications.

Related documentation:

Build logic

The **Palette** contains all of the blocks that you can use in Designer. They are organized into groups according to their type of function. For example, the **Routing** section contains blocks related to routing functionality:

These pages contain more information about the blocks in each category:

- Logic and Control blocks add logical functions to applications, such as assigning variables, changing languages, or directing the application to other blocks.
- User Interaction blocks interact with callers in various ways, such as offering them a list of menu options or invoking a voice or chat bot.
- Business Control blocks control various operational aspects of your site, such as establishing and checking your hours of operation.
- Routing blocks specify how an interaction should be directed when certain conditions are met, such as transferring a customer to an agent.
- Data blocks manage various data-handling functions.
Using the blocks

- External Services blocks manage how an application taps into an external service.
- Reporting blocks manage certain reporting functions within an application, such as starting or stopping an activity or indicating the progress of an application.
- Callback blocks manage how Designer interacts with Genesys Callback services.
- Survey blocks enable you to set up and offer surveys to customers.

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 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.
Using the blocks

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:
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.

Search tools

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. As you type, the items on the page are filtered to show only those items that match the text you've entered. The results are highlighted in the application flow, and you can use the up/down buttons to jump to the next or previous result.

If you click the More Options button, you can also select the Search by Block ID or Enable Deep Search options.

Search by Block ID can be useful if a particular block ID is flagged by Designer Analytics as having an issue. You can toggle the block IDs to be visible in the application flow and then use this search option to quickly locate the block in question.

The Enable Deep Search option enables you to search the entire application flow for a specific value or property. For example, you might search for blocks that contain a certain virtual queue.

This video describes the search tools available in Designer:

Link to video

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.

For information about using ECMAScript in Designer, see ECMAScript Expressions.

Busy Treatments

A busy treatment is a special form of handling that occurs while a customer is waiting to be connected with an agent. For example, music can be played for callers or chat participants can be provided with regular updates about their estimated wait times.

Certain blocks allow you to specify audio files, messages, or self-service type shared modules as busy
treatments. When planning to add a busy treatment to your application, keep in mind that the interaction might be routed immediately or take a very long time. In certain scenarios, you might prefer to use a Shared Module instead of a Play Message block, as you can configure the module to adjust (or skip) the busy treatment messages depending on the Estimated Wait Time (see an example).

Routing blocks

The Route Call, Route Digital, and Route Agent blocks have a Treatments tab where you can specify a busy treatment.

If you choose to add an audio-based or text message treatment, a Play Message block is automatically nested below the routing block. Use this block to select and configure the message 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. Note that you can only select a Self Service shared module.

On the Route Digital block, you can add a chat treatment. This automatically adds a nested Chat Message block below the routing block. You can then use the block properties to set the treatment.

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

Important

- When routing chat interactions, Designer waits for 100 milliseconds before starting busy treatments for chats (i.e. in-queue messages). This significantly improves the accuracy of Estimated Wait Time (EWT) and Position in Queue (PiQ) values that are typically used by these messages when providing updates to customers, but may also result in the application not sending busy treatment chat messages if routing completes almost immediately (i.e. within 100 ms).
- 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 customers while they wait to speak with an agent — and then move on to the routing blocks, all without interrupting the playback to the customer.

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 selected Self Service Shared Module will continue to loop until a new treatment is started, the interaction is routed or terminated, or the Assisted Service phase in which the Start Treatment block was started is exited.

Example of a Shared Module treatment

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:
Using the blocks

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

**Properties - Play Message**

This block is used to play audio messages. These messages can be TTS (Text to Speech), Audio Files (previously uploaded in Audio Resources page, or variables played as TTS.

**Specify prompts to be played**

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td></td>
<td>Transferring. Please be patient. Your estim</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>TTS</td>
<td></td>
<td>EstimatedWaitTime</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>TTS</td>
<td></td>
<td>minutes.</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

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.

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.
Using the blocks

- 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.
Application workflow

Contents

• 1 Overview
• 2 Application streams
  • 2.1 DEV (Development)
  • 2.2 QA (Quality Assurance)
  • 2.3 UAT (User Acceptance Testing)
  • 2.4 LIVE
  • 2.5 LIVE B
• 3 Application builds
• 4 Parallel Test Environment (PTE)
• 5 LIVE B streams
  • 5.1 Sample scenario
Overview

Designer uses a stream-based workflow for developing applications. In this type of model, an application consists of multiple work streams, each of which represents a different stage in the life cycle of the application:

- DEV (Development)
- QA (Quality Assurance)
- UAT (User Accessibility Testing)
- LIVE (Live Production)

When you want to freeze the current state of the application code, you can generate an application build and assign it to a stream. The build can then be promoted to the QA and UAT streams until you are ready to assign the build to LIVE and run it in the production environment.

This type of workflow enables you to have multiple builds of the application running in different streams, completely independent of the others. For example, a developer can work on a build without impacting the build that is assigned to quality assurance or the build that is actively running in production.

This video shows the workflow in action:

[Link to video](#)

Application streams

You can view and manage the streams of an application from the application properties:
There are four application streams:

DEV (Development)
This stream always runs the latest published version of the application code. As such, it is the only stream affected by Designer upgrades.

QA (Quality Assurance)
For builds that require Quality Assurance testing.

UAT (User Acceptance Testing)
For builds that require beta-testing or some other variation of user-based trials.

LIVE
For builds that are actively running in the production environment.

LIVE B
This is an optional stream that can run a second application build in the live production environment.
For more information, see LIVE B streams.

All streams are enabled by default. You can use the Status sliders to enable or disable them as needed.

Application builds

A build is basically a self-contained package of the application code. It has all of the resources it needs to run, so you can assign a build to a stream without worrying about impacts to the original resources or to the other builds.

Builds can be created only after the application has been published. If your application has not yet been published, the Build button remains disabled until the application has been published at least once. When generating a new build, Designer always uses the most recently published version of the application code. If you've made changes to an application that you want to capture in a new build, you must re-publish the application before generating the build.

To create a build, open the application in editing mode and click Build:

Designer automatically adds or increments the version of the build. You must assign the build a Label and, if desired, you can use the Note field to provide any additional information about the build. When you are finished, click Build.
Things to keep in mind when working with builds:

- **Builds are generated from the latest *published* version of the application code.** If you've made changes to an application that you want to capture in a new build, you must re-publish the application before generating the build.

- **Builds are immutable.** Once generated, a build cannot be changed.

- **Rollbacks are permitted.** To perform a rollback, simply assign a previous build to a stream.

- **Each application has a 20-build limit.** Therefore, it's recommended to only generate builds when required. If you exceed this limit, the **Build** button becomes disabled. You can re-enable the button by deleting builds that are no longer needed.

---

**Important**

Application builds must ideally have consistent logic and flow in all application...
streams. In other words, application logic should not be sensitive to the stream it is running on. An application build that exhibits problems in a Live stream will typically be loaded on non-Live streams for troubleshooting. If it changes behavior in the non-Live stream, it may not take the same path it does on the Live stream, which can complicate troubleshooting. It also implies Live stream behavior cannot be tested until the application is made Live, which goes against the streams based workflow that is designed to reduce such risks. It is normal for applications to access different sets of resources in Live and non-Live streams and that aspect is supported by the Parallel Test Environment (PTE) feature.

To manage the builds for an application, use the Manage Builds option when viewing the application properties.

Parallel Test Environment (PTE)

The Parallel Test Environment enables you to use test resources instead of production ones when running an application build in a non-production stream.

To enable this option, go to the Misc tab in the application settings and select Enable for PTE. You can then create copies of certain resources and add a special test_ prefix to the resource name.

Streams that are running non-production builds (DEV, QA, and UAT) will reference the PTE versions of the resources (test_) instead of the ones being used in production. The LIVE build continues to reference the original production resources.

PTE is supported for Business Controls resources (Business Hours, Emergency Flags, Special Days, and Data Tables) and certain configuration resources (Virtual Queues, Agents, Agent Groups, and Skills).

LIVE B streams

You can run an optional LIVE B stream in addition to the existing LIVE stream. This allows you to run a second application in LIVE production mode, which gives you greater flexibility in how you can introduce new application builds into your production environment.

For example, each LIVE stream can have different contact points assigned to it, so one way you could use the LIVE and LIVE B streams is to allocate interactions coming from a certain region to a particular stream, or use the streams to balance (or gradually introduce) the number of interactions being handled by a new production build. In this way, you can implement a form of A/B testing.

Sample scenario

Let's say you wanted to split or balance the number of interactions being handled between the LIVE and LIVE B builds. Typically, you would gradually introduce more interactions from LIVE to LIVE B, which could be done as follows:
Application workflow

1. Assign **Build X** to the LIVE stream.
2. Use the **Manage** button to assign phone numbers to the LIVE stream. At this point, 100% of the interaction load will be handled by **Build X** on the LIVE stream.
3. Next, assign **Build X+1** to the LIVE B stream.
4. Assign some of the phone numbers to the LIVE B stream so that 70% of the interaction load is handled by LIVE and 30% by LIVE B.

   **Tip**
   
   The Transfer from Others button is a great way to easily and quickly move several numbers from one stream to another.

5. Check Designer Analytics to see if there are any issues. If there are no issues, we can move more of the phone numbers from LIVE to LIVE B so that 30% of the interaction load is being handled by LIVE and 70% load by LIVE B.
6. Continue to monitor Designer Analytics for any issues. If everything still looks ok, we can move the remaining phone numbers from LIVE to LIVE B so that 100% of the interaction load is being handled by LIVE B.
7. Assign **Build X+2** to the LIVE stream. This is a new build that we want to test in conjunction with **Build X+1** on LIVE B.
8. Assign some phone numbers to the LIVE stream so that 30% of the interaction load is being handled by our new application build running on LIVE and 70% by LIVE B.
9. Continue to check Designer Analytics for any issues, and adjust and balance the interaction load between the two builds as desired.
Variables

Contents

• 1 Tips
• 2 User Variables
• 3 Securing Variables
• 4 Tracing variables
• 5 System Variables
  • 5.1 List of System Variables
  • 5.2 Setting the MaxTime value
  • 5.3 Setting the ExpirationTime value
  • 5.4 VAR Metrics
• 6 Internal system variables
• 7 Assigning Values to Variables
  • 7.1 Example 1: Simple Assignment
  • 7.2 Example 2: Advanced Scripting
Learn how to use variables in Designer.

**Related documentation:**

---

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 = `abcdef123` or `c123badef`
- Invalid variable names = `123abcdef` or `3abcdef21`

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

Variable values can be captured in a variety of data sources when Designer applications run on the Genesys platform. If a variable contains sensitive data or personally-identifiable information (PII), you can mark a variable as **Secure** to prevent the value from being logged or recorded as plain text in Designer and the Genesys platform.

#### How secure variables work:

- Secure variable values are not captured by application logs.
- If used to store the results of a user input, the user input is masked in platform logs.
- If used to play back a prompt, the prompt message is masked in platform logs.
Secure variables are hidden from view to prevent them from being selected in blocks that record reporting information, such as Call Data, Activity, and Milestone blocks. (See the Warning below for more details.)

Secure variables are not reported in Designer Analytics.

If you secure variables in an application that has already been published, you'll need to re-publish the application for the new settings to take effect.

**Warning**

- **DO NOT** attach sensitive data, such as personally identifiable information (PII) or secure variables, to userdata in the Call Data block. Otherwise, this information is captured by platform logs and reported in Designer Analytics.
- Designer normally hides secure variables to prevent them from being selected in blocks that capture reporting information. However, if you secure a variable that was previously selected in a block as a non-secure variable, Designer cannot remove the variable from the block or prevent its value from being exposed. To protect those values, you must create a new secure variable and re-publish your application.

**Tracing variables**

At this time, Genesys recommends not enabling the Trace option for variables. Instead, use a Debug block.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
<th>Secure</th>
<th>Trace</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>varAppKey</td>
<td>'default'</td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>

**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 with appropriate values when the application session starts and can be used throughout the application, such as the ANI and DNIS. Other system variables, such as Last Milestone, are populated as the application session progresses. For example, when the application starts, the initial value of Last Milestone is an empty string. While the application runs, the Last Milestone value is set to the last milestone that the application reached.
Designer also uses certain *internal* system variables at various stages of the applications session. These are intended only for internal use by Designer. For more information, see Internal Designer system variables.

**Important**

Do not update system variables in the Assisted Service phase while an asynchronous Start Treatment is running. Instead, update system variables *before* the Start Treatment starts or within the Self Service treatment itself.

### List of System Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChatEntryPoint</td>
<td>Holds the point of entry for a chat interaction. Can be used in application logic at runtime to provide alternative processing or to facilitate the use of parallel testing environments.</td>
</tr>
<tr>
<td>DNIS</td>
<td>Specifies the dialed number.</td>
</tr>
<tr>
<td>ANI</td>
<td>The number associated with the calling party.</td>
</tr>
<tr>
<td>MaxTime</td>
<td>Maximum time (in minutes) to keep this session alive. For more information, see Setting the MaxTime value.</td>
</tr>
<tr>
<td>Timezone</td>
<td>The timezone used for this application, unless this value is overridden in other blocks.</td>
</tr>
<tr>
<td>Language</td>
<td>The default language for this application that is used for announcements.</td>
</tr>
<tr>
<td>AppLanguageName</td>
<td>The name of the default language for this application that is used for announcements.</td>
</tr>
<tr>
<td>Persona</td>
<td>The persona to be used for this application. For more information, see Personas.</td>
</tr>
<tr>
<td>RoutingSkills</td>
<td>A set of skills that might be specified in some blocks, such as Menu Option child blocks, that 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.</td>
</tr>
</tbody>
</table>
| 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
**Variables**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>subsequent <strong>Route Call</strong> block, you can enable the <strong>Use system variables RoutingSkills and RoutingVirtualQueue set already in Menu Options</strong> check box to use the value of the <strong>RoutingVirtualQueue</strong> variable.</td>
</tr>
<tr>
<td>EstimatedWaitTime</td>
<td>The estimated wait time for the interaction to be routed to an agent.</td>
</tr>
<tr>
<td>TreatmentIterationCount</td>
<td>Keeps track of how many times a treatment has been executed.</td>
</tr>
<tr>
<td>IVRProfileName</td>
<td>The IVR Profile to associate the interactions with for VAR reporting. The default value is 'auto'.</td>
</tr>
<tr>
<td>GVPTenantID</td>
<td>The tenant to associate the interactions with for VAR reporting. The default value is 'auto'.</td>
</tr>
<tr>
<td>SelectedTarget</td>
<td>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 <a href="mailto:Name@SwitchName.Type">Name@SwitchName.Type</a>.</td>
</tr>
<tr>
<td>SelectedVirtualQueue</td>
<td>The virtual queue that was selected.</td>
</tr>
<tr>
<td></td>
<td>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 <strong>Agent</strong>, <strong>Place</strong>, <strong>Queue</strong>, or <strong>Routing Point</strong>, this variable contains the target. If the desired target type is <strong>Agent Group</strong>, <strong>Place Group</strong>, or <strong>Queue Group</strong>, the function returns the agent, place, or queue from the corresponding group to which the interaction was sent. The target format is <a href="mailto:Name@StatServerName.Type">Name@StatServerName.Type</a>.</td>
</tr>
<tr>
<td>SelectedTargetObject</td>
<td>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: ?::<a href="mailto:SkillExpression@statserver.GA">SkillExpression@statserver.GA</a> or ?GroupName:<a href="mailto:SkillExpression@statserver.GA">SkillExpression@statserver.GA</a>. The target format is <a href="mailto:Name@StatServerName.Type">Name@StatServerName.Type</a>.</td>
</tr>
<tr>
<td>SelectedAgent</td>
<td>This is the Employee ID of the agent to which the interaction was routed.</td>
</tr>
<tr>
<td></td>
<td>(Optional) When present, this is an ECMAScript object that represents a switch access code. The table below show its properties and the corresponding switch access code fields:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Access property</th>
<th>Switch access code field</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>Variable Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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 interaction recording in the <strong>Self Service</strong> 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 interaction's position in queue while waiting to be routed to an agent. This variable is initialized when the application enters a Route Call block. The value is then updated periodically (every x seconds) for as long as the interaction is queued for a target inside the block. The updates stop when the application exits the routing block.</td>
</tr>
<tr>
<td>AgentsTotalSize</td>
<td>The total number of agents that are potentially available. For example, the total number of agents in a specified Agent Group.</td>
</tr>
<tr>
<td>AgentsLoginSize</td>
<td>The number of agents that are actually logged in.</td>
</tr>
<tr>
<td>AppCountry</td>
<td>The country code for this interaction (can be specified by the application).</td>
</tr>
<tr>
<td>AppCountryName</td>
<td>The country name for this interaction (can be specified by the application).</td>
</tr>
<tr>
<td>AppRegion</td>
<td>The region for this interaction (can be specified by the application).</td>
</tr>
<tr>
<td>AppCallType</td>
<td>The type of interaction (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>
</tbody>
</table>
### Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Variable Name</code></td>
<td><code>Description</code></td>
</tr>
<tr>
<td>AppSessionType</td>
<td>The type of the session. The default value is <em>inbound</em> for inbound calls. Survey applications must use the value <em>survey</em>.</td>
</tr>
<tr>
<td>EnableRouteCallRecording</td>
<td>Set to <em>true</em> or <em>false</em> to enable or disable call recording for routed calls in the Assisted Service phase. Leave blank to use platform defaults.</td>
</tr>
<tr>
<td>GmsCallbackServiceName</td>
<td>The GMS Callback Service name.</td>
</tr>
<tr>
<td>GmsCallbackServiceID</td>
<td>The unique identifier that GMS assigns to a scheduled callback.</td>
</tr>
<tr>
<td>survey_sOffer</td>
<td>Set by the Setup Survey block to specify if a survey was offered, setup, or rejected.</td>
</tr>
<tr>
<td>survey_iAgentScore</td>
<td>Holds the user satisfaction score for the agent, if this question is asked by the survey.</td>
</tr>
<tr>
<td>survey_iCompanyScore</td>
<td>Holds the user satisfaction score for the company, if this question is asked by the survey.</td>
</tr>
<tr>
<td>survey_iCallScore</td>
<td>Holds the user satisfaction score for the overall call, if this question is asked by the survey.</td>
</tr>
<tr>
<td>survey_iProductScore</td>
<td>Holds the user satisfaction score for the product, if this question is asked by the survey.</td>
</tr>
<tr>
<td>survey_iRecommendScore</td>
<td>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).</td>
</tr>
<tr>
<td>ApplicationRevisionSerialID</td>
<td>A read-only variable that increments by 1 each time an application is revised.</td>
</tr>
<tr>
<td>ApplicationPath</td>
<td>The absolute path to the application.</td>
</tr>
<tr>
<td>InteractionSource</td>
<td>The source of the interaction. For example, this value could be <em>web</em> (desktop and mobile browsers) or <em>mobile</em> (apps).</td>
</tr>
<tr>
<td>ReferrerURL</td>
<td>The URL that the customer came from.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>The type of browser that the customer is using, e.g. Chrome, Mozilla, Opera, etc.</td>
</tr>
<tr>
<td>UserAgentOS</td>
<td>The type of operating system that the customer is using, e.g. Windows, Mac, etc.</td>
</tr>
<tr>
<td>Interaction</td>
<td>Details about the interaction, e.g. the interaction subject and type.</td>
</tr>
<tr>
<td>Contact</td>
<td>Details about the customer contact (name, phone, etc.).</td>
</tr>
</tbody>
</table>

*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).*
Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>number, email address), stored as a JSON object. For example:</td>
</tr>
<tr>
<td></td>
<td>{&quot;PhoneNumber&quot;: &quot;1234&quot;, &quot;EmailAddress&quot;: &quot;<a href="mailto:name@domain.com">name@domain.com</a>&quot;, &quot;FirstName&quot;: &quot;John&quot;, &quot;LastName&quot;: &quot;Doe&quot;}</td>
</tr>
<tr>
<td>DefaultPartition</td>
<td>The default partition used to provide access control in GIR. This variable can be overridden by settings in the Record block.</td>
</tr>
<tr>
<td>AutoStopInteraction</td>
<td>(Digital only) Specifies whether or not the interaction is to be automatically terminated when the session ends.</td>
</tr>
<tr>
<td></td>
<td>• If set to true, the interaction is automatically terminated just before the application session wraps up.</td>
</tr>
<tr>
<td></td>
<td>• When set to auto (default), the application decides whether to stop the interaction or not based on the interaction media type.</td>
</tr>
<tr>
<td>ChatOfferVQ</td>
<td>The virtual queue that was queried for an Estimated Wait Time (EWT) to determine if chat is to be offered.</td>
</tr>
<tr>
<td>FlowEntryCount</td>
<td>Total number of times (including this run) that the Designer application ran to process this interaction.</td>
</tr>
<tr>
<td>ExpirationTime</td>
<td>Maximum time (in minutes) from when the interaction was first processed to keep the interaction alive. For more information, see Setting the ExpirationTime value.</td>
</tr>
<tr>
<td>isResumedFromParking</td>
<td>This variable indicates how many times (if any) the interaction was parked.</td>
</tr>
<tr>
<td>LanguageForBots</td>
<td>The default language to be used for bots.</td>
</tr>
<tr>
<td>enableSSML</td>
<td>Set as true to enable the interpretation of Speech Synthesis Markup Language (SSML) tags in TTS (Text-to-Speech) prompts. For more information, see Speech Synthesis Markup Language.</td>
</tr>
<tr>
<td>removeSSMLInChat</td>
<td>Set as true to remove SSML tags from chat messages if an omnichannel application is using the same messages for both voice and chat channels. For more information, see Speech Synthesis Markup Language.</td>
</tr>
</tbody>
</table>

Setting the MaxTime value

The MaxTime value represents the maximum length of time (in minutes) that an interaction session can remain active before being automatically terminated. The default setting is 240 minutes, but Genesys strongly recommends that you keep this number higher than the longest possible wait time.
your customers might experience. This is to prevent the session from being terminated before it is completed normally, such as in cases where a customer has requested an ASAP callback. Designer sees the initial call and the subsequent callback as a continuation of the same session, so if the \texttt{MaxTime} expires before the callback is made, that session is lost. (Scheduled callbacks are not affected, as they use a separate session for the callback.)

Recommendations for setting the \texttt{MaxTime} value:

- Use a value that is greater than the maximum wait time on your busiest day.
- Use a value that is also greater than the \texttt{Callback Purge Time} set for ASAP callbacks. (For more information about callback settings, see the Callback Settings Data Table.)

Setting the \texttt{ExpirationTime} value

This system variable is typically more relevant to chat scenarios. It differs from the \texttt{MaxTime} variable in that it specifies the duration of time (in minutes) to keep the interaction active, not the overall application session. Typically, incoming chat interactions are much shorter in duration than voice calls and are processed by multiple Designer applications, back-to-back, which means a single chat interaction can often be associated with multiple sessions.

During the \texttt{Finalize} phase, if Designer detects that an interaction has been active longer than the time specified by the \texttt{ExpirationTime} value, then Designer terminates both the interaction and the session. If the session terminates and the chat is not routed, it gets automatically re-queued, a new session starts, and so on. This could go on forever. Thus, when auto-stop logic is used by the application (set by the \texttt{AutoStopInteraction} system variable), the \texttt{ExpirationTime} system variable triggers Designer to terminate the interaction.

Recommendations for setting the \texttt{ExpirationTime} value:

- For chats, use a value that is greater than the maximum interaction time required when processing the chat with different applications or if the interaction is going to be looped within the same application.

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 \texttt{IVRProfileName} variable (User Data Key: \texttt{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 \texttt{GVPTenantID} variable (User Data Key: \texttt{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.
Internal system variables

During an application session, Designer also adds certain internal system variables at various stages. These internal variables are recorded in Designer Analytics at the end of the application, along with other system and user-defined variables. Thus, it is possible to see new variables in Analytics that are not listed in the Initialize phase block.

Although these variables may appear in variable or call data objects in Session Detail Records (SDR), they are intended only for internal use by Designer and should not be used in blocks for driving application logic. These variables can change or be removed at any time, so attempting to use them in applications can affect application resiliency and cause unexpected behavior. This type of usage is not supported or recommended. If your business operations require new functionality, contact your Genesys representative.

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 into 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.
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`).

```
// assume this data was retrieved from an external system using HTTP REST
varOrderDetails = [
  { "item" : "Laptop bag", quantity : 3, backordered : false },
  { "item" : "Phone charger", quantity : 2, backordered : false },
  { "item" : "Super rare fish", quantity : 1, backordered : true }
];

var i; // a local variable that exists only in this script
varOrdersPrompt = ""; // use a variable defined in Initialize phase
for ( i = 0; i < varOrderDetails.length; --i ) {
  // 3 laptop bags ... give a space between quantity and item name
  varOrdersPrompt += varOrderDetails[ i ].quantity + " + " + varOrderDetails[ i ].item;
  // its odd to hear 2 of phone charger (not chargers) lets fix that
  varOrdersPrompt += varOrderDetails[ i ].quantity > 1 ? 's' : '';
  // add a comma to give TTS a short pause
  if ( i < varOrderDetails.length - 1 ) {
    varOrdersPrompt += "; ";
  }
}
```

The variable will be set to false if an error is thrown during advanced script evaluation, and true otherwise.
Saving and Publishing Your Application

Contents

• 1 Validation issues
• 2 Backup and recovery
  • 2.1 Last Manual Save
  • 2.2 Recover Backup
Learn how to save, validate, and publish your applications.

**Related documentation:**

It is a good idea to manually save your work often, especially after you have made important changes.

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 validation issues.

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.

![Validation Status: ![](icon.png)  Settings  Save Flow  Publish  Build](image.png)

Designer also automatically saves a temporary backup of your work, which is refreshed periodically. If you forget to save your changes, and the editing session ends for some reason (for example, maybe you accidentally closed an active browser tab or were logged out due to inactivity), Designer provides a few backup and recovery options.

**Validation issues**

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

Warnings are displayed in yellow and errors in red. Code generation can complete successfully if there are warnings present, but fails if there are any errors.

Click a warning or error to go to the block containing the issue and address the problem.
Backup and recovery

If you try to edit an application or shared module that has unsaved changes from a previous session, Designer lets you know that 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.

Backup - Recover?

A backup was found from a few seconds ago that is more recent than the last saved copy of this application.

• Last Manual Save: Last Tuesday at 3:48 PM
• Backup: Today at 4:13 PM

Which copy would you like to start from?
(Choosing Last Manual Save deletes the backup.)

You can choose one of the following recovery options:
Last Manual Save

Designer opens the last version that was manually saved to the server. If you select this recovery option, the local backup copy is discarded and any changes that were made after the last manual save are not included in the recovery.

Recover Backup

Designer opens the local backup version, which is recovered from the cache of your browser. You can then click **Save Flow** to save your changes to the server.

**Important**

This backup and recovery feature doesn't work in private browsing modes, such as "incognito" mode in Chrome or "private" mode in Firefox. These modes do not permit data to be stored or retrieved from the browser cache.
Shared Modules

Contents

• 1 Using Shared Modules
  • 1.1 Shared Module blocks
  • 1.2 Routing blocks (as "Busy Treatments")
  • 1.3 Start Treatment block
• 2 Creating a Shared Module
• 3 Importing a Shared Module
• 4 Settings
  • 4.1 General Tab
  • 4.2 DTMF Options Tab
• 5 Other Actions
Shared Modules

• Administrator

Learn how to manage shared modules, which are small pieces of applications that you can use in other applications.

Related documentation:

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 blocks. These templates are read-only and cannot be edited or deleted (but they can be cloned – see Other Actions). 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 Media 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. 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. Note that a cloned module does not inherit the history and published versions of the original module.

**Tip**
If you want to preserve a particular version of a shared module, you can clone it rather than create a new version using the built-in versioning. This allows you to manage and edit the cloned module without affecting the original version.
Shared Modules

• **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 Collections

Contents

• 1 Assigning a media collection to an application
• 2 Managing media collections
  • 2.1 Clone a media collection
  • 2.2 List media collection consumers
  • 2.3 Export a media collection
  • 2.4 Delete a media collection
• 3 Creating a new media collection
  • 3.1 Publishing the collection
• 4 Adding media resources to a collection
  • 4.1 TTS only announcements
  • 4.2 Audio properties
  • 4.3 Message properties
• 5 Import Bulk Audio
• 6 Supported audio formats in browsers
• 7 Exporting collections for offline management
  • 7.1 CSV file fields and descriptions
  • 7.2 Adding new resources to a collection
  • 7.3 Removing resources from a collection
• 8 Speech Synthesis Markup Language (SSML)
  • 8.1 Additional examples
  • 8.2 SSML settings
• 9 Intelligent Prompts
• 10 Music on Hold
• 11 Viewing the history of a resource
Learn about managing media resources and collections.

Related documentation:

From the **Media Collections** page, you can centrally manage the media collections and resources that are being used in your applications and shared modules.

Each collection contains **media resources**. These are TTS (Text-to-Speech) prompts and recorded audio files that you can use in any block that supports the playing of audio or the display of text and standard response messages for chat prompts. For example, a company might have a media collection that contains media resources for each locale in which they operate, with language-specific audio and/or messages for greeting customers, communicating announcements, and offering special promotions.

Designer includes the following media collections:

- **Callback V2 Audio** contains media resources used with Callbacks. You can modify the resources in this collection if you want to customize them for your own requirements. For example, you might want to add additional languages or replace the existing recordings with new ones.

- **Shared Resources** contains media resources that can be used in any application or module. You can make changes to this collection.

- **System Resources** is a read-only collection that is managed by Designer. You cannot make any changes to this collection.

- **Music On Hold** contains the default audio file played for music on hold. You can change the audio that is played, but the collection cannot be modified or deleted.

You can also create your own media collections.

Assigning a media collection to an application

You can assign media collections to applications and shared modules from the Media tab of the application settings. 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.
Managing media collections

You can use the **Action** toolbar to manage 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 changes to the media collection offline 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.
Creating a new media collection

Click Add Media Collection.

Enter a name for the collection. When you are done, click Create and Open to open the new collection and add media resources.

Publishing the 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.

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.

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 to a collection
After you've created a media collection, you can add media resources to it. This video shows an example of how to add a media resource:

**Link to video**

Click the collection's link to view its details. You can then click **Add Media Resource** to add a new media resource to the collection.

**Tip**

If the collection was previously published, you can also export the collection for offline management and then import the files back into Designer.

Give the resource a unique **Name** and click **OK**. Your new resource will appear in the list of resources. Select it to view the the resource properties in the **Media Resource details** section. You can enter an optional **Description** for the resource, or add **Tags** to associate it with other media resources that share the same tags.

Next, click **Manage Languages** and then **Add Language** to select a language to use for this resource. You can then set up the **Audio** and **Message** properties of the resource, as described below.

### TTS only announcements

If this resource is only to be played as TTS (Text-to-Speech), select **This announcement plays TTS only**. Typically, you would select this option if you do not plan to use or upload any recorded audio.

- In the **TTS Text** field, enter the text to be spoken by the TTS engine. You can also use **Speech Synthesis Markup Language** (SSML) in TTS prompts. For more information about SSML, see Speech Synthesis Markup Language (SSML).

- If the TTS only resource is also being used for chats, you can use the **Alternate Text** field to enter a message to be displayed in chats. Otherwise, chats will use the text that was entered in the **TTS Text** field.

### Audio properties

If adding an audio recording, make sure that the **This announcement plays TTS only** box is NOT
selected.

There are a few requirements for audio files:

• The file size must not exceed 10 MB.
• 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 (.) and cannot be longer than 50 characters. 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.

Click the File button to select and upload the recording. A checkmark appears if the file is uploaded successfully. After you have saved the audio resource, you can click the play button to listen to the audio file.

Message properties

If the media resource is also going to be used for chats, you can enter text to display in the Alternate Text field or select a Standard Response message.

When you are finished, click Save. To add more languages for the resource, click Add Language and repeat the above steps.

Import Bulk Audio

The Import Bulk Audio option is only available for collections that have previously been published.
This option enables you to upload a package of media resources to an audio collection. This can be useful if you've exported a collection to make changes—for example, maybe you've added new resources, created new audio recordings, or updated the properties of several resources specified in the associated CSV file—and now you want to refresh the collection with the changes. (To learn more about working with CSV files for media collections, see offline management.)

From the Media Collections page, click the media collection to which you want to import media. Click Import Bulk Audio.

Select the zip file that contains the resources you want to import and click Submit. Designer validates the zip file and reports any errors. You can then review the information provided on the Summary and CSV Rows tabs before confirming the upload. These tabs provide details about the files being imported, such as any changes that Designer has detected between the resources in the existing collection and the resources being imported.

Example of the Summary tab:

Example of the CSV Rows tab:

If everything looks ok, click Confirm Upload. After the media resources are successfully imported to the collection, you can choose to revert the changes or publish the collection.
Remember! Media collections must be published before the changes will take effect in any applications or shared modules that are using the collection.

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.

<table>
<thead>
<tr>
<th>Format</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Internet Explorer</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (IE 9 and later)</td>
<td>Yes</td>
</tr>
<tr>
<td>wav (16 bit mono)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>wav (u law)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>vox</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

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.

Exporting collections for offline management

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark Media Resource For Deletion</td>
<td><strong>Mandatory.</strong> 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.</td>
</tr>
<tr>
<td>Audio Resource ID</td>
<td><strong>DO NOT MODIFY.</strong> 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.</td>
</tr>
<tr>
<td>Audio Resource Name</td>
<td><strong>Mandatory.</strong> Name of the audio resource. You can modify this value.</td>
</tr>
<tr>
<td>Audio Resource Description</td>
<td>Description of the audio resource. You can modify this value.</td>
</tr>
<tr>
<td>Audio Resource Tags</td>
<td>Tags used to group the audio resources for easy identification. Tags should always be enclosed within double-quotes (&quot;tag&quot;). Multiple tags can be separated by a comma, but kept within the double-quotes: &quot;tag1, tag2, tag3&quot;.</td>
</tr>
<tr>
<td>Mark Language For Deletion</td>
<td><strong>Mandatory.</strong> 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.</td>
</tr>
<tr>
<td>Play Only Text</td>
<td><strong>Mandatory.</strong> 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.</td>
</tr>
<tr>
<td>Audio Resource Language</td>
<td><strong>Mandatory.</strong> Indicates the language that the audio file/text supports. The language can be any one of the languages supported by Designer.</td>
</tr>
<tr>
<td>Audio File Name/Text to Play (if Play only Text is true)</td>
<td><strong>Mandatory.</strong> 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.</td>
</tr>
<tr>
<td>Text to Be Played</td>
<td>Indicates the text to be played if there is issue playing the audio file. Enter the TTS text to be played.</td>
</tr>
</tbody>
</table>
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.

**Important**

If you make changes to a media collection, you must re-publish it for the changes to take effect in the applications or shared modules that are using it.

Speech Synthesis Markup Language (SSML)

Designer supports the use of **Speech Synthesis Markup Language** (SSML) in TTS prompts. You can use these tags to control how TTS phrases are spoken by the TTS engine, such as reading back a prompt in a specific type of format or placing an emphasis on certain words or syllables. The tags use a syntax similar to HTML tags, in that they enclose the text value to be speech formatted in an opening and closing tag.

For example, the following prompt uses SSML tags to specify a *date* format:

1960/01/02

The TTS engine reads this back as "January second, nineteen sixty."

Or, if you specify an *ordinal* format:

You are the 5 customer in the queue.

The TTS engine reads this back as "You are the fifth customer in the queue."

When enabled for an application, SSML handling is in effect for any block that plays or consumes TTS messages, such as the Bot (includes responses received from the bot service), Menu, and Play Message blocks.

To learn more about SSML standards, see the Speech Synthesis Markup Language page at the World Wide Web Consortium (W3C) website.
Additional examples

These examples are provided for illustrative purposes only, to demonstrate some of the capabilities of SSML markup. TTS service providers might not support all SSML tags, so you'll need to check with your provider to confirm which ones you can use.

<table>
<thead>
<tr>
<th>SSML</th>
<th>TTS Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234 Main Street., Northport, NY 11768</td>
<td>&quot;Twelve thirty-four Main Street, Northport, New York, one one seven six eight.&quot; (spoken at 80% speed)</td>
</tr>
<tr>
<td>Let me think about it. You are 39 years old.</td>
<td>&quot;Let me think about it. You are (pause for 1.5 seconds) thirty-nine years old.&quot;</td>
</tr>
<tr>
<td>This date: 3/6/02 sounds the same as this date: 03/06/2002</td>
<td>&quot;This date March sixth two thousand two sounds the same as this date March sixth two thousand two.&quot; (Note that the different date formats are rendered the same when spoken.)</td>
</tr>
<tr>
<td>1742 Main St., Salem, OR 96607</td>
<td>&quot;Seventeen forty-two Main Street, Salem, Oregon, nine six six zero seven.&quot;</td>
</tr>
<tr>
<td>10:30PM on 1960/01/02</td>
<td>&quot;Ten thirty P M on January second, nineteen sixty.&quot;</td>
</tr>
<tr>
<td>banana</td>
<td>&quot;B A N A N A.&quot; (The TTS engine spells out the word.)</td>
</tr>
</tbody>
</table>

SSML settings

You can make changes to the SSML settings in the application system variables. For new applications, SSML handling is enabled by default, but you can set the enableSSML variable to true (enabled) or false (disabled), as desired.

If your application is using the same TTS prompts and messages for both voice and chat channels, you can set the removeSSMLInChat system variable to true. This removes the SSML tags from prompts when they are displayed to customers during chats.

Important

Designer does not validate SSML tags. If you are using them in your prompts, make sure that you are using the correct syntax and that the tags are supported by your TTS services provider.

Intelligent Prompts
Designer can intelligently convert a number into an item (such as a day or currency) and then read it to the customer using human-sounding audio. For more information, see Intelligent prompts.

Music on Hold

The **Music On Hold** media collection contains an audio resource (**Music on Hold**) that plays as the default audio when customers are placed on hold. To change the music that is played, you can edit the properties of this resource to upload a new audio file to play.

To do this, click **Music On Hold** to view the collection. Select the **Music on Hold** audio resource to display its properties and click **Manage Languages**. Use the browse button to select and upload a new audio file:

Designer automatically stores the uploaded file as **music_on_hold**. However, it keeps the same extension as the uploaded file.

Viewing the history of a resource
When viewing the details of a media resource, you can use the History button to view a list of historical events for the resource.

The history view displays a list of events for the selected item, such as each time the resource was viewed, edited, or published. You can then drill-down into each event to see more details.

For more information, see History view.
Personas

Contents

• 1 What is a persona?
• 2 Enabling personas
  • 2.1 Disable personas
• 3 Selecting a persona
• 4 Changing the persona
• 5 Managing personas
• 6 What happens if the language changes?
Choosing a persona for your application can provide customers with a more personalized experience.

**Related documentation:**

A persona is an artificial voice that you can use to control Text-to-Speech (TTS) services in your applications. They are designed to sound more natural and conversational than the robotic voice that is typically associated with IVRs and can portray certain personality characteristics that customers might find more appealing or appropriate. For example, you might choose a more formal-sounding persona for one particular type of scenario and a cheerful one for another.

The following are examples of persona types (provided for illustrative purposes only):

- Samantha (female, 30-40s)
- Tom (male, 30-40s)
- Gabriela (female, 20-30s)
- Michael (male, 20-30s)
- Diane (female, 40-50s)
- David (male, 40-50s)

Depending on how you want to use personas, you can set the related system variable to apply a persona globally to the entire application or use a Change Persona block to switch to a different persona at a specific point in the application.

**Important**

The personas that are available can vary, depending on your environment and how you intend to use them. Check with your Genesys representative to confirm which personas are available and supported for your deployment.
Enabling personas

Personas are not enabled by default. To enable personas for an application, go to the Persona tab in the application settings and select the **Enable Persona** check box.

After personas are enabled, you can select a persona for the application.

Disable personas

If you want to disable personas for an application, clear the **Enable Persona** check box. Any personas specified by the **Persona** system variable or **Change Persona** blocks will then be ignored and all TTS services will revert to the default voice that is provided by that service.

Selecting a persona

To set a persona that will be used globally throughout the entire application, go to the System Variables tab in the **Initialize** phase. Find the **Persona** variable and select the type you want from the list of available personas.

For example:
The persona that you select will be used by any blocks that are using Text-to-Speech (TTS) services, such as Play Message, User Input, Menu, Bot, Route Call, and Route Agent. If desired, you can change the persona at various points in the application.

If personas are enabled for the application and you don’t select a specific persona (or choose use default from the list) the default persona is used. Typically, the default persona is “Samantha”, but this can vary depending on your configuration.

Changing the persona

You can use a Change Persona block to dynamically change the persona. You can add this block to the Self Service or Assisted Service phase of your application or use it in a shared module. The new persona takes effect with the next prompt that is played.

Switching personas is useful when you want to use certain personas for specific situations. For example, you might want to use a more formal persona when dealing with sensitive customer issues, or switch to different personas depending on the customer segment being served (“Gold" customers get one type of persona, "Blue" ones get another).

For more information about using this block, see the Change Persona block page.

Managing personas

You can view a list of available personas on the Media Resources > Personas page. From this page, you can select a persona to view additional details about it, such as the Language, TTS Provider, and TTS Voice Name.
What happens if the language changes?

If the language of the application is changed (such as by a Change Language block), the TTS voice continues to use the selected persona. For example, if you are using the persona of a young female and the language changes from English to Spanish, the voice will still use that type of persona. If there is no voice available that matches the selected persona, an alternate voice or TTS service is used for that language.
Intelligent prompts

Contents

• 1 About intelligent prompts
• 2 Using intelligent prompts
• 3 Available languages
• 4 Limitations
  • 4.1 Special characters
Intelligent prompts

Intelligent prompts are pre-recorded audio files that sound similar to natural human speech. They can be arranged to be spoken in a way that adheres to regional standards, such as when presenting a currency denomination or a date.

Related documentation:

About intelligent prompts

As an alternative to TTS, Intelligent Prompts play back pre-recorded audio files that sound more similar to natural human speech. They are also dynamic, in that you can arrange them to be presented in a format that is specific to a certain region, such as a currency denomination or a date standard. For example, you might need to specify Euros instead of Dollars, arrange the prompt so that the day is spoken before the month, or have a number spoken as an ordinal, where a number such as 21 is read back as "twenty-first".

Intelligent prompts can be used with the following types of items:
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetical characters</td>
<td>For example, &quot;P&quot;.</td>
</tr>
<tr>
<td>Cardinals</td>
<td>Speaks a number as a quantity. For example, <strong>1234</strong> would be spoken as <strong>&quot;one thousand, two hundred, thirty-four.&quot;</strong></td>
</tr>
<tr>
<td>Currencies</td>
<td>Speaks a number as a specified currency denomination. For example, in the U.S. English (US-en) locale where dollars ($) are used, <strong>11234</strong> would be spoken as <strong>&quot;eleven thousand, two hundred and thirty-four dollars.&quot;</strong></td>
</tr>
<tr>
<td>Days</td>
<td>Speaks the specified day of the week. Days are numbered from 0-7, starting with Sunday. For example, <strong>1</strong> would be spoken as <strong>&quot;Monday&quot;.</strong></td>
</tr>
<tr>
<td>Months</td>
<td>Speaks the specified month. Months are numbered from 1-12, starting with January. For example, <strong>5</strong> would be spoken as <strong>&quot;May&quot;.</strong></td>
</tr>
<tr>
<td>Ordinals</td>
<td>Speaks a number as an ordinal. For example, <strong>21</strong> would be spoken as <strong>&quot;twenty-first&quot;.</strong></td>
</tr>
</tbody>
</table>
Using intelligent prompts

Intelligent prompts can be used in any block that supports prompts, such as Play Message, User Input, and Menu blocks. When setting up a prompt, select **Intelligent Prompt** as the **Type** and indicate how the prompt is to be played by selecting the appropriate option from the **Play As** menu (alphanumeric, ordinal, cardinal, etc.).

Available languages

The languages that are available for playing intelligent prompts are listed on the Language tab in the Application Settings. From this tab, you can select the languages to be used for playing intelligent prompts.

Intelligent prompts are available for the following languages:

<table>
<thead>
<tr>
<th>Language</th>
<th>ISO Country Code</th>
<th>Voice Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese (Hong Kong)</td>
<td>zh-hk</td>
<td>Female</td>
</tr>
<tr>
<td>English (Australia)</td>
<td>en-AU</td>
<td>Female</td>
</tr>
<tr>
<td>English (Great Britain)</td>
<td>en-GB</td>
<td>Female</td>
</tr>
<tr>
<td>English (United States)</td>
<td>en-US</td>
<td>Female</td>
</tr>
<tr>
<td>French (Canada)</td>
<td>fr-CA</td>
<td>Female</td>
</tr>
<tr>
<td>Italian (Italy)</td>
<td>it-IT</td>
<td>Female</td>
</tr>
<tr>
<td>Punjabi (India)</td>
<td>pn-IN</td>
<td>Female</td>
</tr>
<tr>
<td>Spanish (Mexico)</td>
<td>es-MX</td>
<td>Female</td>
</tr>
</tbody>
</table>
Important

These language packs are included with all versions of Designer. Additional languages may be available for your deployment. Check with your Genesys representative for details.

If multiple languages are selected in the Application Settings, Designer uses the overall language setting in the application, such as the language set by the Language setting in the system variables or a Change Language block, to determine which language to use for playing intelligent prompts. If Designer is not able to play an intelligent prompt in the preferred language, it plays the prompt in the default fallback language. Unless changed, the default fallback language is en-US (English, United States).

Limitations

There are certain limitations to be aware of when using intelligent prompts:

Special characters

Special characters (including spaces) in intelligent prompts are not always interpreted the same by all languages. For example, one language might speak the words "special character" when playing back a special character in a prompt, while another language might ignore certain special characters completely. This can cause unexpected results when using the same prompts in multiple languages. Genesys recommends testing your prompts in each language you are using to ensure they are spoken as intended.
Digital Resources

Contents

• 1 Creating Message Collections and Message Resources
• 2 History view
Digital Resources

- Administrator

Set up and manage digital resources for your applications, such as standard responses.

**Related documentation:**

**Digital 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.

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 immediately across all applications that are consuming that resource.

(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.

Selecting a message resource enables you to view and make changes to the resource properties:

You can:
Digital Resources

- Add a **Description** for the resource.
- Add some **Tags** to associate the resource with similar resources.
- Add a **Language** to create a custom **Text** message or select a standard response **Message** from the Standard Response repository.
- Use the **History** button to view the history of the digital resource.

**History view**

To view the history of a digital resource, select it and click **History**.

The history view shows you a list of each time the resource was viewed, edited, or published, the user who made the change, and the new and previous value of any properties that were changed.

The results can be sorted, filtered (for example, you can use the buttons to see only the history for a day, week, or month, or manually enter a specific start and end date), searched, and exported to a file.

If you double-click an event row, an audit window opens that displays details for that particular event.
Manage personally identifiable information

Contents

• 1 See how it works
• 2 Create or edit a rule
• 3 Test a rule
• 4 Change a rule's status
• 5 Reorder a rule
• 6 Copy a rule
• 7 Delete a rule
• 8 Best practices
• 9 Regular expression examples
• 10 Partial masking
Learn how to assign rules and actions to incoming messages, emails, and workitems to protect your customers' private information.

**Related documentation:**

Use PII Rules Management to assign rules to all incoming messages in chat, SMS, WhatsApp, Facebook (private and public messages), Twitter (private and public messages), email, and workitems. These rules use regular expressions to detect private information that you can mask with the replacement text of your choice. For example, you could mask a phone number like this: (###)###-####

**Important**

Privacy rules are not applied to outgoing messages, such as those sent by a Designer application, an agent, or a bot.

You can create rules for any alphanumeric string that follows a defined pattern represented by a regular expression. When the rule finds a match, it masks the data with a custom-defined string. Here are some common private fields that you can match with a rule:

- Account number
- Credit card number
- Phone number
- Email address
- Date of birth

Access PII Rules Management under the *Digital Resources* menu.
See how it works
The following video shows how PII rules can be used to protect private information:

Link to video

Create or edit a rule

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Order</th>
<th>Scope</th>
<th>Media</th>
<th>Replace with</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-digit PIN</td>
<td></td>
<td>21</td>
<td>HISTORY</td>
<td>WORKITEM</td>
<td>$***</td>
</tr>
<tr>
<td>Credit Card</td>
<td></td>
<td>11</td>
<td>ALL</td>
<td>EMAIL</td>
<td>$****<strong><strong><strong><strong>-</strong>**-</strong></strong>-</strong>**</td>
</tr>
<tr>
<td>DOB</td>
<td></td>
<td>31</td>
<td>AGENT</td>
<td>MESSAGING</td>
<td>$****<strong><strong><strong><strong>-</strong>**-</strong></strong>-</strong>**</td>
</tr>
<tr>
<td>Phone number</td>
<td></td>
<td>500</td>
<td>AGENT</td>
<td>EMAIL, MESSAGING, WORKITEM</td>
<td>(###)###-<strong><strong>-</strong></strong></td>
</tr>
</tbody>
</table>

PII Rules Management

Add New Rule
Manage personally identifiable information

New Rule

Name *
Phone number

Description
Find a phone number and replace all digits.

Media *
MESSENGING, EMAIL, WORKITEM

Order *
500

Scope *
- Mask everywhere inside the system (ALL)
- Mask for specific cases: when being shown to agent (AGENT)
  - when being saved to system for history purposes (HISTORY)

Regular expression *
/ (\d{3})\d{3}\d{4} / g ~

Replace with
(###)#####

Test message
My phone number is (425)555-1212

Test result
My phone number is (###)#####
When you create a rule, a **New Rule** view opens on the right side of the page. To begin, give your rule a name and a brief description.

Next, choose the types of **Media** for which the rule should apply. Note: The MESSAGING type represents all chat, SMS, and social media messages.

The **Order** determines the sequence in which rules are applied to the message, starting with the lowest number. The default is 500, but you can choose any value between 0 and 9999. You can also update the order after the rule is saved.

**Important**

Genesys does not recommend assigning the same order value to multiple rules in the same scope.

Choose when to apply the rule to the message by selecting the **Scope**. If you mask the data everywhere inside the system (ALL), then the rule is applied right after the user sends a message during the interaction. The data is permanently removed for the message and can't be retrieved later. **If the media is an email or workitem, you must choose this option.**

If you mask for specific cases, then the system treats the data as follows:

- **AGENT** - The rule is applied when the user's message displays to the agent. The data is saved in the system and can be retrieved later.
- **HISTORY** - The rule is applied when the message is saved to the system for historical purposes. With asynchronous messages, personal information in the active segment of communication is only visible to the agents who own the active interaction. Personal information in the previous communication segments is hidden.

The **Regular Expression** determines which information to replace in the message. You must use ECMAScript syntax to define the regular expression. The regular expression text area also has a dropdown to set flags for global, multi-line, or case-insensitive searching.

The **Replace with** string is ***** by default. You can change this value to any pattern that makes sense for your use case. For example, it could be (###)###-#### to mask a phone number.

PII Rules Management also supports partial masking.
Finally, don't forget to test your rule before saving.

Test a rule

PII Rules Management includes a testing feature you can use to confirm that the rule is working as expected.

To test, open the rule and enter a Test message. When you click Test, the Test result field shows how your rule handles the test message.

You can adjust your regular expression and replacement text as needed, just remember to click Save when you have finished.

Change a rule's status

After you create a rule, you can manage whether it is enabled with the Status switch ( ). The switch is off by default, which means the rule is not applied to messages. Any changes to the status take effect immediately.

Reorder a rule

After you create a rule, you can change the order in which it is applied. Hover over the Order cell and increase or decrease the value in increments of 10.

Copy a rule

You can copy an existing rule with the Copy button ( ). This opens the New Rule view with the copied rule information. The name of the rule includes "_N", where "N" is the number of the copy. For example: Rule_1, Rule_2, and so on.
Delete a rule

To delete a rule, just click the delete icon (人才队伍图标). If you just want to disable the rule temporarily, consider changing the rule's status instead.

Best practices

Here are some key best practices for managing personally identifiable information:

1. Create the minimum number of privacy rules. It's difficult to analyze and maintain many privacy rules.
2. Create the strictest possible regular expressions. For example, if you want to mask a credit card number, make sure you're not masking any 16-digit number. Your regular expression should be as specific as possible to the data you are masking, such as:
   - Start with a new word
   - End a word
   - Have specific delimiters
   - Have specific numbers in particular positions

Regular expression examples

Here are examples of some common regular expressions:

**Credit Card (Visa, MasterCard, and Discover Card only)**

```regex
(?:^|(?!\s))\d(3|5\d{2}|6\d{4})(?:\D\d{3}){2}\d{4}(?:\s\d\d\s\d\d\s\d\d\d{4}|\D\d\d\d\s\d{4})
```

**SSN (Social Security Number - U.S. only)**

```regex
(?!000|666|9)\d{3}(?!00)\d{2}(?!000)\d{4}(?!0$)|(?=\Da-zA-Z()...?)
```

Partial masking
You can partially mask sensitive information by using capturing groups in your regex. Use $ in the Replace with field for any group you want to exclude from the mask.

To exclude more than one capturing group, each subsequent group must have at least one replacement symbol before the group in the Replace with field.
Manage standard responses

Contents

• 1 Plan a category tree
• 2 Build a category tree
  • 2.1 Create or edit a category
  • 2.2 Copy and paste a category
  • 2.3 Delete a category
• 3 Create or edit a standard response
• 4 Copy a standard response
• 5 Delete a standard response
• 6 Search the category tree
• 7 Characters allowed in names
• 8 Make images accessible
Standard responses are a way for your organization to define a set of prewritten responses. Agents or automated processes can send these responses to customers in email, chat, SMS, WhatsApp, Facebook, and Twitter interactions.

A standard response is a text template with keyword placeholders, called field codes, that the system replaces with appropriate text for the recipient. This results in a standardized, yet custom message that is crafted according to the situation and customer. Genesys uses standard responses as suggestions for agents, acknowledgments, and autoresponses.

Agents can further personalize the standard response by it in their desktop before sending the message to the customer. See the Agent Workspace Agent’s Guide for more information about how agents use standard responses.

This topic covers the steps to create standard responses using Standard Responses Management. At a high level, you must:

1. Plan your category tree. A category tree provides the framework for organizing your standard responses.
2. Build your category tree.
3. Create your standard responses.
4. Use Field Code Management to create field codes, then add them to your standard responses. Field codes are the keyword placeholders you can use to personalize standard responses.

Access Standard Responses Management under the Digital Resources menu.
Plan a category tree

A category tree is a way to organize your library of standard responses into a structure that is easy to access and manage.

A category tree consists of one or more "root categories" (units of knowledge), each of which can have subcategories or even sub-subcategories. These categories and subcategories contain the standard responses.

It's important to plan your category tree ahead of time. You can edit it as needed, but a little thought now will save time and frustration later. When planning, think about the root categories and subcategories that make sense in your business.

Let's use an online store as an example. Our store website can handle both chat and email interactions from customers. We want to send different responses depending on the interaction type, so let's make root categories for Chat and Email. Those root categories need subcategories for the most common reasons customers contact us: Orders, Returns, and Rewards. Since Orders is a large topic, let's break it down into sub-subcategories: Order Status and Order History. Our site supports French and English, which means we must actually create two category trees - one for French and one for English.

Here's how our English Category Tree Plan looks:
The way you organize your categories is subjective and depends on the needs of your business. For example, we could have made root categories for Orders, Returns, and Rewards. Then populated those categories with standard responses that are named by interaction type (Return Label Chat, Return Label Email, and so on).

Here's how this Alternate Category Tree Plan looks:

You can have a category with no standard responses - its purpose is to group other categories together. However, a standard response must belong to a category.

Category membership is inherited. For example, Chat includes Orders, and Orders includes Order Status and Order History. Then Chat also includes Order Status and Order History.

Build a category tree

Now that planning is complete, it's time to build the category tree. In these instructions, you build the category tree first and then add standard responses, but you can also add standard responses as you build your category tree.
Create or edit a category

**Important**

Category folders with the star icon (★) cannot be edited. These categories are created and managed in Workload Manager for Intelligent Workload Distribution and Engage Cloud Email and are critical to proper functionality. While you can't edit the category, you can still create, edit, and delete standard responses within the category.

To begin, choose a language from the drop-down - remember, your category tree is specific to the language.

**Next,** click **Add** to create a root category. In the category editor, give your category a name and helpful description. After you save, the standard response table displays. You can use this table to add standard responses now or continue defining your tree.

**Important**

The category name can consist of no more than 64 alphanumeric characters supported in UTF-8. For additional valid characters see Characters allowed in names.

Once you have a root category, select its menu (▼) and add a child category. You set up child categories the same way as root categories. First, define the category in the editor and then add standard responses as needed.
Continue adding categories until your tree is complete. As you work with your category tree, you can adjust the layout by using the copy and paste feature or deleting a category.

Copy and paste a category

If you need to change your category tree structure, you can use the category menu (>>) to **copy** a category.

To save a copied category as a root category, click **paste** in the main sidebar.

To save a copied category as a sub-category, click the menu (>>) of the new parent category and select **paste**.

The copied category is saved as "_N", where "N" is the number of the copy. For example: Chat_1, Chat_2, and so on.

You can also use the category menu (>>) to **cut** a category, and then paste it as previously described. Choose **clear buffer** to clear the contents from your computer's clipboard.
Delete a category

You can delete a category by choosing **Delete** in the category menu. If you delete a category, all of its subcategories and standard responses are also deleted.

Create or edit a standard response

You can create a standard response in two ways: the category menu or edit the category and add a standard response to the table.

Either way triggers the standard response view, where you can set the name and description, content, attachments, and properties.

**Important**

The standard response name can consist of no more than 128 alphanumeric characters supported in UTF-8. For additional valid characters see Characters allowed in names.
Manage standard responses

### Content

**Order acknowledgement**

**Subject**

Thanks!

Plain text part

```
Thanks for your order, {{$ Contact.FullName $}}!
```

**HTML part**

```
9 WORDS POWERED BY TINY
```

**Attachments**

**Properties**

Valid from 01/06/2021 1:53 PM

---

### Attachments

**Order acknowledgement**

**Thanks!**

**Attachments**

Drop files to attach or browse

**Properties**

Valid from 01/06/2021 1:53 PM

---

### Properties

**Order acknowledgement**

**Thanks!**

**Attachments**

**Acknowledgement**

- Will be used automatically

**Response**

- Cannot be used

**Suggestions**

- Cannot be used

**Start Date**

2021-01-06, 01:53 PM

- Will Never Expire
First, define the name and description for your standard response.

Expand the Content section and add the subject and body for the standard response. The body can be in either plain text or HTML, depending on the type of standard response. For example, use plain text for SMS responses and HTML for email.

HTML mode includes an integrated editor. Most of the buttons provide commonly used editing functionality, including the option to insert a link to an image. If you aren’t sure what a button does, hover over it to open a tooltip.

You can insert field codes to customize the standard response. Field codes are supported for both plain text and HTML standard responses. If the field code you need is not in the list, you can create it using Field Codes Management. Delimit field codes by using . If you type a field code directly into the body of a standard response, then you must enter the delimiters yourself. If you select from the list of field codes, then the delimiters are added automatically.

To add an attachment, save your progress. The save process enables the Attachments section, where you can add or remove an attachment by drag-and-drop or browsing your computer. When adding an attachment, make sure to consider the channel type you are targeting for the standard response and align the attachment type with the channel. Different channel types might have different limitations on what is acceptable.

In the Properties section, define whether the standard response is used manually, automatically, or not at all for these scenarios:

- Acknowledgment—The standard response is sent through Designer to acknowledge the receipt of a message. For example: “Thank you for your email. One of our agents will reply to you in the next 24 hours.”
- Response—The standard response is sent by agents to an incoming interaction. It can also be used by Designer if you can be certain the customer’s message is answered in this response.
- Suggestions—The standard response is offered to agents as suggested wording to use in their own replies to interactions.

Finally, set a start date (today’s date by default) and specify whether the standard response expires.

If a standard response’s expiration date is reached, it has the following effects:

- Designer cannot use the standard response in a new or modified application.
- If a Designer application saved a standard response after the expiration date, Genesys does not send the standard response. Instead, Genesys returns an error message.

When your standard response is ready, click the check box to approve. Approved standard responses are identified with a check mark icon (✔). You can only use approved responses in Designer applications.

Copy a standard response

You can copy responses from the standard responses table in a category. Just click **Copy** (COPY) for the standard response you want to copy and then paste it to the current category.
Manage standard responses

To copy a standard response to a different category, choose Paste from the category's menu.

Important
The copy/paste functionality does not copy any existing standard response history. This history might be present in a standard response if it was originally created using the eServices Manager Plug-in.

Delete a standard response

You can delete standard responses from the responses table view in a category. Just click Delete for the standard response you want to remove.
Manage standard responses

Search the category tree

To locate a category or standard response, use the search box above the category tree. The category tree filters automatically based on the search term.

To search for standard responses in a category, use the search box above the responses table. The results filter automatically based on the search term.

Characters allowed in names

<table>
<thead>
<tr>
<th>Name</th>
<th>Character</th>
<th>Name</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyphen</td>
<td>-</td>
<td>Exclamation point</td>
<td>!</td>
</tr>
<tr>
<td>Number sign, pound</td>
<td>#</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>Caret</td>
<td>^</td>
<td>Asterisk</td>
<td>*</td>
</tr>
<tr>
<td>Underscore</td>
<td>_</td>
<td>Curly brackets</td>
<td>{ }</td>
</tr>
<tr>
<td>Angle brackets</td>
<td></td>
<td>Period, full stop</td>
<td>.</td>
</tr>
<tr>
<td>Backslash</td>
<td>\</td>
<td>Parentheses</td>
<td>( )</td>
</tr>
<tr>
<td>Question mark</td>
<td>?</td>
<td>Space</td>
<td></td>
</tr>
</tbody>
</table>
You can make images in your HTML-based standard responses accessible by adding alternative text. In HTML mode, click the **Insert/Edit Image** button to add your image and include the alternative text in the **Alternative description** field.

You can also add the "alt" attribute directly to the `<img>` tag when working in the source view (☞).
Manage field codes

Contents

• 1 Create or edit a field code
• 2 Use variables in field codes
  • 2.1 Use your own data
• 3 Use formulas in field codes
  • 3.1 Field Code Syntax
  • 3.2 HTML in Field Codes
  • 3.3 Operator Precedence
  • 3.4 Functions
  • 3.5 Using Objects
• 4 Examples
• 5 Copy a field code
• 6 Delete a field code
• 7 Search for a field code
• 8 Characters allowed in names
Field codes are the most complex and powerful aspect of standard responses. With field codes, you can create standard responses that are automatically personalized when they are used.

For example, you can use the field code `Contact.FirstName` in a response beginning Dear Contact.FirstName, which you send to dozens of recipients. In each message, `Contact.FirstName` is replaced by the first name of the addressee of the message (the contact) as listed in the Universal Contact Server database.

Field Codes Management enables you to create a wide range of field codes types. The codes can range from simple - similar to a Mail Merge word processor feature - to complex field codes that include multiple objects, formulas, and constants.

Once you create a field code, you can use it in multiple standard responses.

The interface for creating field codes is simple. Creating useful field codes requires a deeper understanding of how they are constructed. Create or edit a field code provides step-by-step instructions, with links to detailed reference information at the relevant places.

Access Field Codes Management under the Digital Resources menu.

Create or edit a field code
When you create a new field code, the **New Field Code** view opens on the right side of the page. To begin, give your field code a name and brief description.

**Important**

The field code name can consist of no more than 64 alphanumeric characters supported in UTF-8. For additional valid characters see Characters allowed in names. The following characters combinations are not allowed: `**`.

Choose system or custom variables to insert into the **Text** field, and add any other necessary text. See Use variables in field codes for details about variables and how to use them. If the custom variable you need is not in the list, you can create it using Custom Variables Management.

For detailed help with constructing the field code, see Use formulas in field codes.

**See Examples for more about using complex field codes.**

**Use variables in field codes**
The ability to access interaction data is perhaps the most frequent use of field codes. Although field code formulas can be complicated, many just retrieve a single piece of data, such as a contact’s name.

You access Universal Contact Server data using predefined variables, called "system variables."

These variables access three predefined objects. Each object has a name and a set of properties. Let's use Contact.FirstName as an example. Contact is an object and FirstName is one of its properties. The system variable Contact.FirstName retrieves the value of the FirstName property of the Contact object.

Similarly, there is a system variable for each object+property pair. You can use the following objects in field codes:

• Agent object - Information about the agent, such as name and signature.
• Contact object - Contact details such as name, address, and phone number.
• Interaction object - Information about the interaction, such as created date, to and from address, and attached data.

Use your own data

It is possible to incorporate data that you keep external to Universal Contact Server into your standard responses (including automated responses). This data could include case numbers, account information, and so on. Remember that attached data always consists of key-value pairs.

Follow this two-step process to use external data:

1. Retrieve the external information and add it to the interaction as attached data. One place to do this is in a Designer application.
2. Now that you have attached the data to the interaction, use the AttachedData property of the Interaction object to access the data. The AttachedData property requires one argument, which is the key name. The result of the following formula is the value associated with the OrderStatus attached-data key: Interaction.AttachedData("OrderStatus")

Use formulas in field codes

In addition to system variables such as Contact.FirstName, field codes contain formulas. This section provides an outline of formula usage. Find details about these topics in the Field Codes Reference Guide.

Field code formulas are similar to formulas in other applications, such as Microsoft Excel.

A formula is a sequence of one or more operands (such as numbers and text strings), separated by operators (such as + and -).

For example, in the following formula, 2 and 3 are operands and + is an operator: 2 + 3

Operands can be values that do not change (constants), or values that vary based on the context. In the previous formula, all the operands are constants, so the formula always evaluates to 5. The next formula evaluates to a different value for each agent who uses it: Agent.Signature
Field Code Syntax

To summarize field code syntax:

- Enclose alphabetic strings, whether constants in formulas or elsewhere in a field code, in double quotes.
- Numeric constants require no special treatment.
- Use special characters for some purposes. For example, for your field code to render with a line break, you cannot just type a carriage return. Instead, you must insert the expression `\n`. See the Escape Sequences table in the Field Codes Reference Guide.

HTML in Field Codes

With special configuration, field codes can contain HTML markup; for example, you could have a field code that uses a custom variable with a default value of:

Sam Agent
Acme Products
29 Exterior Blvd
Springfield, CX 09090

To enable this, you must use the Java property `-Dsr1-field-code-allow-html=true`, in one of the following ways:

- Add it to the JavaArgs section of ContactServerDriver.ini
- Add it as an argument to the startup command line in contactServer.sh.

Operator Precedence

If you use more than one operator in a formula, the order in which they are evaluated depends on their relative precedence. Higher precedence operators are evaluated first. For example, multiplication (*) has a higher precedence than addition (+), so that the formula below evaluates to 14, not 20:

2 + 3 * 4

You can use parentheses to override the default precedence. The formula below evaluates to 20:

(2 + 3) * 4

For a complete list of operators and their relative precedence, see Operator precedence in the Field Codes Reference Guide.

Data Types

Operands of several different types may appear in formulas:

- Number
Each data type behaves differently in formulas, and the operators have different meanings when you use them with different data types. For example, the + operator means “add” when used with numbers, but “concatenate” (paste together) when used with strings. This formula evaluates to *Uncle Sam Wants You*:

"Uncle Sam " + "Wants You"

You can't use some operators with some data types at all. For example, you cannot use the multiplication (*) operator on two strings.

Genesys converts all formulas, regardless of their final data type, to strings before merging them into your standard response. This conversion follows a set of default rules that depend on the data type. For example, the default rules for numbers round them off to integers. This formula inserts 2 into your standard response, even though the real result is 2.25:

9 / 4

You can use the Text function or format operator (:) to override the default formatting. Either of the following formulas inserts 2.25 into your standard response:

Text(9 / 4, ".##")

(9 / 4):".##"

For a detailed list of data types and how you can use them, see Data types in the *Field Codes Reference Guide*.

**Functions**

When composing formulas, you can use many built-in functions. **Functions** are predefined formulas that perform calculations using values, called **arguments**, which you supply. To use a function, write its name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis.

Function arguments can be of any data type, although individual functions can place restrictions on their arguments. Function arguments can be constants or formulas. The Length function, for example, takes a single string argument and returns its length in characters. This formula evaluates to 13:

Length("Hello, world!")

As another example, the Date function takes individual date components (year, month, day, and so on), and constructs a date/time value. The formula below evaluates to 1965-11-23 09:03:10:

Date(1965, 11, 23, 9, 3, 10)

Functions can act as arguments to other functions. The WeekdayName function takes a single date/
time argument and returns the day of the week as a string. The formula below evaluates to Tuesday:

\[ \text{WeekdayName(Date(1965, 11, 23, 9, 3, 10))} \]

This formula evaluates to 7:

\[ \text{Length(WeekdayName(Date(1965, 11, 23, 9, 3, 10)))} \]

For detailed descriptions of all available functions, see the following topics in the Field Codes Reference Guide:

- String functions
- Date and time functions
- Type conversion
- Mathematical functions
- Miscellaneous functions

Using Objects

All object/property pairs are available in the system and custom variables menus.

Object properties can be of any data type. Agent.FullName, for example, is a string, but Interaction.DateCreated is a date/time.

The data type of an object property can even be another object. For example, Contact.EmailAddresses yields another object called a ContactEmailAddressList. You can access the properties of the resulting object by entering a period (.), followed by the property name, just as before. The formula below evaluates to the number of email addresses assigned to the contact:

\[ \text{Contact.EmailAddresses.Count} \]

Some object properties require arguments just as functions do. For these properties, write the arguments, enclosed in parentheses after the property name, just as before.

For example, the ContactEmailAddressList object has a property named Exists. You can use this property test whether a contact has a particular email address. The data type of this property is Boolean (true/false), and it takes one argument, the email address to test.

\[ \text{Contact.EmailAddresses.Exists("sam@acme.com")} \]

For detailed descriptions of all objects and their properties, see the following topics in the Field Codes Reference Guide:

- Agent object
- Contact object
- Interaction object
Examples

The following is an example of a complex field code:

If (Time() - Interaction.DateCreated > 14, “Please accept our apologies for not having replied sooner. ”, “”)

This field code inserts a tardiness apology if more than 14 days have elapsed since the interaction first entered the system. It uses the function If, which has these properties:

• Its syntax is If (Boolean, TrueResult, FalseResult)
• If Boolean evaluates to True, it returns the second argument.
• If Boolean evaluates to False, it returns the third argument.

In this example, the three arguments of If are as follows:

1. Time() - Interaction.DateCreated > 14 A formula that returns True if the difference between the date created and the current system time is more than 14 days. (A mathematical operation on dates returns a result in days.)
2. “Please accept our apologies for not having replied sooner.” A text string apologizing for tardiness, inserted if the formula evaluates to True.
3. The null string: if the reply is not late (the formula evaluates to False), nothing is inserted in it.

Copy a field code

You can copy an existing field code with the Copy button (COPY). This opens the New Field Code view with the copied information. The name of the field includes "_N", where "N" is the number of the copy. For example: Greeting_1, Greeting_2, and so on.

Delete a field code

To delete a field code, just click the delete icon (DELETE).

Search for a field code

To search for a field code, use the search box above the table. The results filter automatically based on the text you enter in the search box.
### Characters allowed in names

<table>
<thead>
<tr>
<th>Name</th>
<th>Character</th>
<th>Name</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyphen</td>
<td>-</td>
<td>Exclamation point</td>
<td>!</td>
</tr>
<tr>
<td>Number sign, pound</td>
<td>#</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>Caret</td>
<td>^</td>
<td>Asterisk</td>
<td>*</td>
</tr>
<tr>
<td>Underscore</td>
<td>_</td>
<td>Curly brackets</td>
<td>{ }</td>
</tr>
<tr>
<td>Angle brackets</td>
<td></td>
<td>Period, full stop</td>
<td>.</td>
</tr>
<tr>
<td>Backslash</td>
<td>\</td>
<td>Parentheses</td>
<td>( )</td>
</tr>
<tr>
<td>Question mark</td>
<td>?</td>
<td>Space</td>
<td></td>
</tr>
<tr>
<td>At sign</td>
<td>@</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Manage custom variables

Contents

• 1 See how it works
• 2 Create or edit a custom variable
• 3 Copy a custom variable
• 4 Delete a custom variable
• 5 Search for a custom variable
Manage custom variables

- Administrator

Learn how to create and edit custom variables to use in standard responses.

Related documentation:

Use Custom Variables Management to create custom variables to use in field codes. Custom variables have the following properties:

- Their values are assigned by strategy objects.
- Standard responses that use field codes containing custom variables must have the usage type Acknowledgment.

Access Custom Variables Management under the Digital Resources menu.

See how it works

Watch the following video to see how you can manage custom variables for field codes in Designer:

Link to video

Create or edit a custom variable
When you create a new custom variable, the New Custom Variable view opens on the right side of the page. To begin, give your variable a name and brief description.

The custom variable name can consist of no more than 64 alphanumeric characters (a-z, A-Z, 0-9) and underscore.

Choose the Type of custom variable and add a default value that corresponds to the type. If you choose DateTime, use the calendar widget to select a default date and time or enter your own value in this format: "yyyy-MM-ddTHH:mm:ss.SSSZ".

Copy a custom variable

You can copy an existing variable with the Copy button (✓). This opens the New Custom Variable view with the copied information. The name of the variable includes "_N", where "N" is the number of the copy. For example: CV_DateTime_1, CV_DateTime_2, and so on.
Delete a custom variable

To delete a custom variable, just click the delete icon (삭제).

Search for a custom variable

To search for a custom variable, use the search box above the table. The results filter automatically based on the text you enter in the search box.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV_DateTime</td>
<td>DateTime</td>
<td>2021-01-06T14:43:01.700Z</td>
</tr>
<tr>
<td>CV_String</td>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>CV_Int</td>
<td>Integer</td>
<td>101</td>
</tr>
<tr>
<td>CV_Name</td>
<td>String</td>
<td>Default Value</td>
</tr>
</tbody>
</table>
Speech Grammars

Contents

• 1 Creating a speech grammar
• 2 Using speech grammars
A **speech grammar** defines the list of phrases or options that the caller can input when they use your application.

**Related documentation:**

Click **Speech Grammars** 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 (note that the size of the uploaded file cannot exceed 10 MB).

After the speech grammar file has uploaded, you can click the **Show Contents** button ( ) 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.

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.
Bot Registry

Contents

• 1 Before you start
  • 1.1 Supported bot service providers
• 2 Adding a bot resource to the registry
• 3 Adding a Google Dialogflow CX bot to the registry
• 4 Adding a Google Dialogflow ES bot to the registry
  • 4.1 Wait Conversation End
• 5 Adding an Amazon Lex bot to the registry
Learn how to add external bot services to the **Designer Bot Registry** so you can use them in your applications.

**Related documentation:**

You can access the Designer **Bot Registry** from the **Grammars & Bots** menu.

---

**Before you start**

The Bot Registry does not enable you to create bots. Rather, it stores information about bots that you've configured with a supported bot service provider.

Before you can add a bot resource to the registry, you'll need to set it up with your bot service provider. During registration, you might need to provide certain account details, such as URLs, user IDs, or passwords that Designer requires to access the bot. If you are not sure where to get the information required to register the bot, check with your bot service provider.

After you've added a bot to the registry, you can start using it by adding a Bot Block to your application.

**Supported bot service providers**

---

**Important**

Some bot services may not be available in all deployments. Check with your Genesys representative to confirm which bot services are supported in your environment.
Designer currently supports the following bot types:

- Google Dialogflow CX (can be enabled upon request)
- Google Dialogflow ES
- Amazon Lex

Adding a bot resource to the registry

**Link to video**

Click **Add Bot Definition** and enter the details for your bot. You can then test your bot's connection and save it to the registry. Watch the video to see a quick demonstration of how to add a bot resource to Designer.

For more information about adding specific bot types, see:

- Adding a Google Dialogflow CX bot
- Adding a Google Dialogflow ES bot
- Adding an Amazon Lex bot

Adding a Google Dialogflow CX bot to the registry

For Google **Dialogflow CX** bots, you can import a JSON file that contains the settings for your bot resource. You can export this file from your Dialogflow CX account page. If you need help doing this, refer to the Dialogflow documentation.

Click **Add Bot Definition** and enter a **Name** for the bot (check valid naming characters for a list of approved characters you can use when naming a bot resource). For the **Type**, select **Dialogflow CX** from the list of supported bot services.
Click **Choose file** to select the JSON file that contains the setting for your bot and click **Open**. Designer imports the settings for your bot and displays the bot properties. You must also manually enter the following details for your Dialogflow CX bot service to work with Designer:

- **Location** - the geographical location (i.e. region ID) of your CX bot agent. Examples: us-east1, europe-west1, asia-south1.

- **Agent ID** - the agent ID of your CX bot.

- **Environment ID** (Optional) If you have created different versions and environments in your bot settings, you can use this field to enter the ID of an environment that corresponds to a specific version of the flow. You can find this ID in the bot settings (under **Manage > Testing & Deployment > Environments**) by checking the URL of the selected environment:

  ![Environment ID](image)

  If left blank, Designer uses the latest version of the draft agent that is deployed to the default environment.

Here is an example of the properties for a Dialogflow CX bot:
Example of the properties for a Dialogflow CX bot. Click for larger view.

After you’ve entered the required settings, click **Test Connection** to check if Designer can communicate with it. If the test fails, check your bot details and try again. When you are finished, click **Save** to add it to the registry. You can now use the bot in your applications by adding a Bot Block to your application.

Adding a Google Dialogflow ES bot to the registry

For Google **Dialogflow ES** bots, you can import a JSON file that contains all of the required settings for your bot resource. You can export this file from your Dialogflow account page. If you need help doing this, refer to the Dialogflow documentation.

Click **Add Bot Definition** and enter a **Name** for the bot (check valid naming characters for a list of approved characters you can use when naming a bot resource). For the **Type**, select **Dialogflow**.
Click **Choose file** to select the JSON file that contains the setting for your bot and click **Open**. Designer imports the settings for your bot and displays the bot properties.

**Wait Conversation End**

You can also select an optional setting called **Wait Conversation End**. When enabled, the Bot Block recognizes intents that are marked as end-of-conversation intents in the bot.

Here is an example of the properties for a Dialogflow ES bot:

Example of the settings for a Dialogflow ES bot. Click for larger view.

After you've entered the settings, click **Test Connection** to check if Designer can communicate with it. If the test fails, check your bot details and try again. When you are finished, click **Save** to add it to the registry. You can now use the bot in your applications by adding a Bot Block to your application.
Adding an Amazon Lex bot to the registry

Click **Add Bot Definition** and enter a **Name** for the bot (check valid naming characters for a list of approved characters you can use when naming a bot resource). For the **Type**, select **LEX** and click **Submit**.

For Amazon Lex bots, you'll need to provide the following details:

- **Bot Name** – The name of the bot in Lex.
- **Bot Alias** – The name of the alias that the bot was published to. (An *alias* points to a specific version of your Lex bot.)
- **Region** – The region the bot is located in. This corresponds to the region indicated in your Amazon Web Services (AWS) console (e.g. us-east-1, us-west-2).
- **Access Key ID** and **Secret Access Key** – These credentials are created and managed from AWS Identity and Access Management (IAM) and require the appropriate permissions to interact with Lex.

Here is an example of the settings for a LEX bot:

![Example of the settings for a LEX bot. Click for larger view.](image)

After you've entered the required settings, click **Test Connection** to check if Designer can communicate with it. If the test fails, check your bot details and try again. When you are finished, click **Save** to add it to the registry. You can now use the bot in your applications by adding a Bot Block to your application.
Business Controls

Contents

• 1 Emergency Flags
• 2 Business Hours
• 3 Special Days
• 4 Data Tables
• 5 Callback Settings
Business Controls

The Business Controls pages help you to manage resources and settings that are specific to your site, such as Business Hours, Emergency Flags, Special Days, and Data Tables.

Related documentation:

Any changes made on the Business Controls 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 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

Contents

• 1 Adding an Emergency Flag
• 2 Activating or Deactivating an Emergency Flag
• 3 Checking for Emergency Flags
Emergency Flags

Learn how to create and manage Emergency Flags that your application can use to handle emergency conditions.

**Related documentation:**

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 interactions if the check was done prior to activation.
Special Days

Contents

- 1 Add a Special Day
Set up and manage Special Days (such as holidays) that you can reference in your applications.

**Related documentation:**

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.
Business Hours

Contents

• 1 Specify open and closed times
  • 1.1 Important information for Callback
• 2 Specify exceptions
• 3 Time Zone (for Callback only)
  • 3.1 Example
Set up and manage the hours of operation for your business.

**Related documentation:**

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. Once Business Hours are set, you can use them in your applications with a Business Hours block.

**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.

**Important information for Callback**

If you are using this business hours entry for Callback (i.e. in the Callback Settings Data Table), you must also specify the **Time Zone** setting. See Time Zone, below, for more information.

**Specify exceptions**
You might want to specify alternate 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 exception.

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

```
[Image of exception table]
```

When you add an exception (or override), it is enabled by default. You can clear the check box to disable it.

**Important**

- You must select a **Special Day** for each exception. Otherwise, the exceptions will not take effect.
- Even if a **Special Day** exception is enabled, it won't take effect unless the **Follow Overrides Defined Below** option is selected.
- If the current day is a Special Day and there are multiple matching exceptions defined for it, Designer uses only the first exception that matches the specified conditions and ignores any others.

**Time Zone (for Callback only)**

If you are using Callback, select the appropriate **Time Zone** to use for booking scheduled callbacks. This setting ensures that customers who request callbacks are offered time slots that correspond to the time zone of the business hours that are being used for callbacks, and not the time zone of the Designer application (as set in the system variables).

The **Time Zone** setting only applies when determining the available time slots for booking scheduled callbacks. It does not affect any other callback features or settings, such as determining if the business is open or closed for offering callbacks or when to initiate **Immediate Blackout**. These calculations are always based on the time zone that is specified in the System Variables of the
Business Hours

Designer application.

Example

Let's say our application is running in the Eastern Daylight Time (EDT) zone (i.e. America/New_York), as specified in the System Variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timezone</td>
<td>(GMT-5:00) America/New_York</td>
<td>TimeZone for this application used unless it is overridden in other blocks.</td>
</tr>
</tbody>
</table>

However, our contact center is located in California and is open from Monday to Friday, 9 am to 5 pm (Pacific Daylight Time). In this case, we would specify the business hours accordingly:

<table>
<thead>
<tr>
<th>Day</th>
<th>Start Time</th>
<th>End Time</th>
<th>No End Time</th>
<th>Open All Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Then, for the Time Zone, we would select America/Los_Angeles to use for booking scheduled callbacks. This setting will override the time zone that is currently set in the Designer application and be used for determining the available callback times:

Time Zone - Required only for Callback

(GMT-8:00) America/Los_Angeles

If a customer calls and requests a callback, they are offered an available time slot that is based on the America/Los_Angeles time zone. All other features and settings continue to use the America/New_York time zone that was set for the application in the System Variables.
Data Tables

Contents

• 1 What is a data table?
• 2 When should I use a data table?
• 3 How do I add a data table to my application?
• 4 Search bar
• 5 Creating a new data table
  • 5.1 Defining the column settings
  • 5.2 Finishing up
• 6 Editing a data table
  • 6.1 Adding or removing rows
  • 6.2 Changing column settings
  • 6.3 Changing data table values
  • 6.4 Searching the data table
  • 6.5 Reviewing your changes
  • 6.6 Saving the data table
  • 6.7 Publishing the data table
• 7 Viewing the history
• 8 Exporting and importing data tables
  • 8.1 Export
  • 8.2 Import
• 9 Editing exported data tables in Excel
Learn about data tables and how you can use them in your applications.

**Related documentation:**

Go to **Business Controls > Data Tables** to view and manage your data tables.

What is a data table?

A data table contains values that can be read by a Designer application. It has rows and columns and looks similar to a spreadsheet (in fact, you can even export a data table and edit it in a program such as Microsoft Excel), but operates more like a database. Each data table has at least one primary key column, which Designer uses to lookup and retrieve (or store) a value from the table.

When should I use a data table?

Data tables are useful when you want an application to refer to values that are stored outside of the application, or if you want Designer to update values without actually changing them in the application.

For example, you might want customers from a specific region to receive a different welcome message than other customers. Or you might want to specify additional routing handling based on a particular condition being met, such as the business hours for that day or a customer’s ID.

Also, certain Business Controls (such as Business Hours and Special Days) have corresponding data types, which means you can add them to a data table and then create and edit their values directly from within the data table.
How do I add a data table to my application?

When you want Designer to reference a data table, just add a Data Table block to the application flow and select the data table you want from the list. To learn more about using this block, see the Data Table block page.

Search bar

The **Data Tables** page has a standard Search bar:

![Search bar](image)

As you start to type in the **Search** box, Designer filters the list of data tables to display any matches.

**Important**

To optimize system performance, the search function on the **Data Tables** page only looks for matches in data table **Names**, **IDs**, and **Tags**; it does not search the contents of individual data tables. By special request, the search can be extended to include data table contents, but with the understanding that this can have a negative impact on system performance. If you choose to enable this extended search capability in your environment, Genesys strongly recommends that you consider this choice to be permanent, and to not request this change if optimal system performance is preferred.

Designer does allow you to search certain columns of a data table and filter for a match when it is open for editing. For more information, see Searching the data table.

Creating a new data table

To create a new data table, click **Add Data Table** and enter a unique name for it.

You can then click **Create** to save the new data table and return to the main **Data Tables** page, or **Create and Open** to save it and start configuring the data table properties.

Some things to keep in mind when planning or creating a 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.
A data table is not intended to be used as a full-scale database—there is a limit as to how much data can be stored. Focus on data that is frequently updated or critical to your operations.

As an example, let's create a data table that tells the application to perform some special handling for a voice call based on a dialed number (DNIS). We’ll call it **Joules Coulomb Data Table**:

![Creating new Data Table](image)

Click **Create and Open** to create the table and open it for editing. Our table does not have a schema (structure) yet, so Designer asks us to create one, which we’ll do in the next step.

**Defining the column settings**

When a data table is first opened for editing, Designer asks you to define the structure, or *schema*.

Click **Manage Schema** to define the **Column Settings**. (If the data table already has a schema defined, you can get to the column settings by clicking **Column Settings** when the data table is open for editing.)

In this example, we want to use the **DNIS** as the *lookup key*. The lookup key is the column that holds the value that Designer will search for when referencing the data table. We’ll create a column called **Dialed Number** and select it as a *Key* column. We can then define the additional properties for this column, as follows:

![Column Settings](image)

**Key?**

Indicates if this column is a key column used to look up a row of values. In the example above, we
want our column to be a key, so we've selected this option. **Note:** You can only select this option after you have entered values for the other column properties.

Clearly define the lookup keys, as these are important for searching for (and locating) the target data.

---

**Tip**

**Composite Keys:** You can select more than one **Key** column to create a composite lookup key. But note that if a key column is used for a lookup query and contains a blank or invalid value, Designer ignores it when returning the results. In effect, Designer treats it as if it is a wildcard, not as a "no match".

---

**Important**

You can't use a value of "0" (zero) in a **numeric** or **integer** key column. This causes a validation error.

---

**Column**

The name of a column to add to your data table. In the above example, we've entered DNIS.

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.

---

**Important**

Column names must include only alphanumeric characters, must not include any spaces, and must not start with numerals. Note that column names are not validated by the user interface and using any reserved characters in column names may cause unexpected behaviour.

---

**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). In the above example, we've entered Dialed Number (DNIS).

---

**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, we've selected **string**.
Tip
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.

Optional Restrictions
For certain data types, you might want to specify any special restrictions. For example, you could limit a string value to a certain number of characters.

Finishing up
Add and define any additional columns that are needed. When done, click Save. You now have a data table with a key column of Dialed Number. You can go to Editing data tables for information about how to make changes to the data table, such as modifying its settings and adding rows and values.

Here is our example table with some additional columns added:

When you are ready to use the data table in an application, click Publish Table and use a Data Table block to add it to your application.

Editing a data table
To open a data table for viewing or editing, go to Business Controls > Data Tables and click the link for the data table you want to edit. (If another user has the data table open for editing, you will only be able to view it in read-only mode.)
**Tip**
Some browsers might display a script error or seem to freeze when opening large data tables. This is normal, and usually only temporary. You can let the script continue or wait for the browser to finish loading the data table.

From the editing mode, you can

- add or remove rows
- change column settings
- update values
- locate a specific value or row
- save and/or publish the data table

**Don’t forget!** You can save your changes, but they won’t take effect until the data table is published.

**Adding or 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.

**Tip**
For best system performance, limit the number of rows to 1000 and the total size of the data table to no more than 10,000 cells.

**Changing column settings**

Click **Column Settings** to 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** types or whether to enable **Enforce non-overlapping dates** for **datetimerange** types.

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.

---

**Changing data table values**

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 displayed in blue:

---

**Updating Business Hours and Special Days**

You can change these directly in the data table. If you click on a Business Hours or Special Days value, you can select a different item or create and add a new one.

One thing to keep in mind — if you add new **Business Hours** or **Special Days** to a data table, the new business object is **local** to that data table. In other words, it can only be used by the data table it was created in. It won't appear in the global **Special Days** list and it won't be available to select in other data tables.

Local business objects appear in **bold** with a (local) label:
Searching the data table

There are a couple of ways you can quickly locate a specific value or row:

- Each column header has a search box. As soon as you start typing, Designer shows only those rows that contain a match to what you have entered.
- The Row Count box at the bottom of the page lets you jump directly to the specified row number.

Reviewing your changes

At the bottom of the data table, a tracker displays how many 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 the data table

When you are ready to commit your changes, click **Save Table**. Make sure to review your changes! After you click **Save Table**, the changes can't be undone.

Designer validates your changes and lets you know if there are any errors.

Tip

During the save operation, you might see some values (particularly for **Business Hours** or **Special Days**) suddenly change to **N/A**. This is just temporary, and the correct values will re-appear after the save completes.

Don't forget! Saving a data table only preserves the changes you have made. To activate the changes, you must publish the data table.

Publishing the data table

Click **Publish Table** to publish (i.e. activate) the data table in the live production environment. Click **Confirm Publish** to start the publishing process.

If you are publishing changes to an existing data table, Designer also lists the differences between the previously published version of the data table and the one you are publishing now. You can review these changes before clicking **Confirm Publish**.
After you publish the data table, the applications that reference it have access to the latest changes.

**Important**

Designer may not always be able to display a complete list of detected changes on the confirmation page. However, you can still proceed with publishing the data table.

**Viewing the history**

Click **View History** in the **Actions** toolbar to open the history view (i.e. audit log) for a data table.

The history view shows you a list of each time the data table was viewed, edited, or published, the user who made the change, and the new and previous value of any properties that were changed.

You can use the buttons on the page to view the results for a specific time period (for example, **last 1W** to see the results for previous week), or use the date fields to specify a custom date range.

Results can be sorted or searched and you can use the **Export** button to export the results as a CSV file.

You can drill-down further into each results item by double-clicking it. This opens an audit window that displays additional details for that particular event.

**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**
• DO NOT open an exported data table in Excel before reading the information contained in Editing exported data files in Excel. This will help you to avoid any unexpected data changes in your file.
• When importing the edited CSV file, do not change the column headers in Designer. The headers of the data table columns 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

<table>
<thead>
<tr>
<th>LOB</th>
<th>Greetings_Ac</th>
<th>Announcements_1</th>
<th>Announcements_2</th>
<th>Special_Days_1</th>
<th>Special_Days_2</th>
<th>Business_Hrs</th>
<th>Closed_Ann</th>
<th>Auto_Attend</th>
<th>Announcements_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS_IN</td>
<td>false</td>
<td>AZ_IN_Open</td>
<td>AS_IN_Spec</td>
<td>AS_IN_VM_Gr</td>
<td>AS_IN</td>
<td>AS_IN_VM_Gr</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AZ_IN</td>
<td>true</td>
<td>AZ_IN_Open</td>
<td>AZ_IN_Spec</td>
<td>AZ_IN_Holiday</td>
<td>AZ_IN</td>
<td>AZ_IN_Closed</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AZ_Rev_IN</td>
<td>true</td>
<td>AZ_Rev_Open</td>
<td>AZ_Rev_IN_Spec</td>
<td>AZ_Rev_IN_Holiday</td>
<td>AZ_Rev_IN</td>
<td>false</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSV File

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

```
AZ_IN,true,,dbo63d70-37d6-11e6-a888-e53ed8cf09b,AZ_IN_Spec,2165b9f0-37d7-11e6-a888-e53ed8cf09b,AZ_IN,0509d4e0-37d7-11e6-a888-e53ed8cf09b,false,,,VQ_AZ_IN,AZ_IN,30,AZ_IN,30,,,true,EstimatedWaitTime,78e070d0-37d7-11e6-a888-e53ed8cf09b,a38864a0-37d7-11e6-a888-e53ed8cf09b,,true,false,,Arizona_IN_transaction,e5ff1b00-37d5-11e6-a888-e53ed8cf09b
```

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:

![Data Table in Microsoft Excel](image)

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**

To import a CSV file into a data table, locate it in the Data Tables list and click **Import** in the **Actions** column.

**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 a populated data table, make sure that the CSV file and the data table use the same table headers. If the headers do not match, Designer displays an error.

Editing exported data tables in Excel

If you use Microsoft Excel to edit a CSV file, the program might re-format some of the data without any indication that these changes are being made. This can cause unexpected issues when the file is later imported back into Designer.

You can use the following guidelines to safely open a CSV file in Excel for editing. (The steps as described might differ slightly from your version of Excel. If you get stuck, you can use the help tool in Excel to find more information about how to perform that step in your version.)

- In Excel, open a new blank workbook. Go to the Data tab and select Get Data > From Text.

  **Tip**

  If you only have an option for combined Text/CSV, you can enable the legacy wizard for importing a text file by going to File > Options > Data and enabling the From Text (Legacy) import wizard. Once enabled, you can then select Get Data > Legacy Wizards > From Text (Legacy) to open your file.

- Browse to the CSV file for the data table you want to open and select Import.
- In the Import Wizard, choose Delimited as the original data type and click Next.
Select **Comma** as the **Delimiter**. Make sure to deselect any other checkboxes that are selected. Click **Next**.
In the **Data preview**, the first column should now be highlighted. Hold down the **Shift** key and click on the last column to select and highlight all columns. You might need to scroll to the right, depending on how many columns your data table has.

With all columns selected, select **Text** as the **Column data format**. Click **Finish**.
If prompted, in the **Import Data** dialog, select **New worksheet** for the data destination and click **OK**.

The file is now opened safely for editing. When you are ready to save the file and import it back into Designer, make sure to save it as a **CSV** (comma-delimited) type of file. This is the only type of file that Designer allows for data table imports.
Data Tables
Admin Resources

Contents

• 1 Partitions
Learn how manage Administrator settings for Designer.

**Related documentation:**

- Administrator

**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.
Partition-Based Access Control

Contents

• 1 Overview
  • 1.1 For example...
• 2 Partitions tab
• 3 Users tab
• 4 Resources tab
Partition-Based Access Control

Administrator

PBAC enables Designer Administrators to control which resources users have access to.

Related documentation:

Overview

Use the settings on the Partitions page (under the Admin Resources tab) to manage partitions, resources, and users. To view this page, you must be assigned to the Designer Administrator role in Designer. For more information about roles in Designer, see Permissions and Access.

As a Designer Administrator, you can 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 Media and Digital Resources.

For each partition, you can then select the users who will belong to it. Users will only be able to see and manage those resources that are assigned to the partitions they belong to. What a user can then do with an accessible resource (view, edit, and so on) is determined by their Designer role. Each user's PBAC details are stored in their Workspace settings and retrieved during login.

Tip

You can also use PBAC to control which resources are displayed to users in drop-down lists when setting up columns with enumeration data types in Data Tables.

Watch this video to learn how PBAC works:

Link to video

Watch this 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 of their appropriate partitions:

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

**Remember:** 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 diagram 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.

---

**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**:
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:
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 some system-based data tables (such as CALLBACK_SETTINGS and NUMBER_VALIDATION_CONFIGURATIONS). All users have access to these resources.
Logic and Control Blocks

Contents

• 1 Assign Variables
• 2 Change Language
• 3 Go To
• 4 Return
• 5 Segmentation
• 6 Shared Module
• 7 Terminate Call
• 8 Terminate (Digital only)
Use these blocks to add logic functions to an application.

**Related documentation:**

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*
Logic and Control Blocks

**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

Contents

• 1 Assignments tab
  • 1.1 Using functions in expressions
• 2 Sort Function tab
• 3 Advanced Scripting tab
Assign Variables Block

- Administrator

Use the Assign Variables block to assign a new value or expression to user variables.

**Related documentation:**

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.

User variables are specified 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
Properties - Assign JSON Array

This block can assign values of expressions to variables. Define a variable in the Initialize phase or block and select it in this block to assign it values or results of ECMAScript expressions. You can also call ECMAScript utility functions, such as sorting an array, and provide an input to be run through the function.

Assignments

String values must be surrounded by single quotes.

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.
- If you are using a function in an **Expression** and are on Designer Private Edition or Multicloud CX hosted on Azure, you must include a directive. See Using functions in expressions for more information.

**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.

Using functions in expressions

For Designer deployments on Private Edition or Genesys Multicloud CX hosted on Azure, the following directive must be added when using a function in expressions:

```plaintext
/*pragma:ors_await*/
```

Examples:

```plaintext
/*pragma:ors_await*/ _genesys.queue.priorityTuning(InteractionID, true, false, false);

/*pragma:ors_await*/ _genesys.queue.setInteractionAge(InteractionID, true);
```
Important
Example expressions are provided for illustrative purposes only. Due to certain platform dependencies and limitations, not all functions are supported when used in Designer.

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`
- `yyyy-MM-ddTHH:mm:ss[+-]HHmm`

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. For more information, see ECMAScript Expressions.

Properties - Advanced Assign Variables

This block can assign values of expressions to variables. Define a variable in the Initialize phase or block and select it in this block to assign it values or results of ECMAScript expressions. You can also call ECMAScript utility functions, such as sorting an array, and provide an input to be run through the function.

Assign Variables Block

Assignments Sort Function

<table>
<thead>
<tr>
<th>Input Array</th>
<th>Key to Sort By</th>
<th>Sort Order</th>
<th>Key Data Type</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>entry_time</td>
<td>Ascendin</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>Second</td>
<td></td>
<td></td>
<td>-- auto --</td>
<td></td>
</tr>
<tr>
<td>Third</td>
<td></td>
<td></td>
<td>-- auto --</td>
<td></td>
</tr>
</tbody>
</table>

jsonArray

Write your ECMAScript here. Be careful - don't burn yourself!

```javascript
let jason = { "arg1": argOne, "arg2": argTwo }
```
Contents

• 1 Using this 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:
Properties - Change Language

This block changes the language of the IVR, and also the preferred routing language.

Use variables

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:

Properties - Initialize

This block or phase is typically used to setup variables for the application and initialize them. Assign blocks can be used to calculate expressions and assign their results to variables in this phase.

**User Variables**  **System Variables**

Specify User Variables. String values must be surrounded by single quotes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Private</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageName</td>
<td>‘ar-SA’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LanguageName1</td>
<td>‘bg-BG’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DisplayName</td>
<td>‘Arabic (Saudi Arabia)’</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DisplayName1</td>
<td>‘bulgarian’</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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

Language Name

Display Name
Classify Block

Contents

• 1 Categories tab
• 2 Results tab
You can use this block to classify a digital interaction based on certain keywords in the content.

**Related documentation:**

---

**Categories tab**

Use the settings on this tab to select the categories that Designer will use to classify the interaction and the confidence threshold to be applied.

![Properties - Classify](image)

If you use individual categories, segment blocks are created. When the application runs, the segment associated with the highest scored category is executed (similar to a Segmentation block).

**Results tab**

Specify the variables in which to store the **outcome** and **results** of the classification operation and the **name** and **relevancy** of the most relevant category.
Click-to-Call-In Match Block

Contents

• 1 Using this block
• 2 Match Criteria tab
• 3 Routing Parameters
• 4 Results tab
Click-to-Call-In Match Block

- Administrator

Use this block to add Click-to-Call-In Match functionality to your application.

**Related documentation:**

You can use this block to support the **Click-to-Call-In Match** feature. In a typical Click-to-Call-In Match scenario, a caller dials a contact center and provides some additional information during the Self Service phase, such as the reason why they are calling. They are then given a phone number to dial along with a special access code.

The application uses this block to compare the caller's information with the details of the original request, such as the access code they entered, the number they called from, or the number they dialed. If a match is found, the additional information that was submitted with the original request is returned to the block.

Depending on the matching criteria used, the match block then generates all combinations for the output as segmentation branches, where each segmentation has its own **Navigation** tab that specifies where to route the interaction.

**Using this block**

Typically, you would add a User Input block ahead of this block to prompt the caller for their access code and specify the variable that will hold it. Then use the **Click-to-Call-In Match** block to specify the matching criteria and the variable that will hold the returned data, if a match is made.

For example, if all three criteria are used, this creates eight possible outcomes (or segmentation branches), including an outcome where **Nothing Matched**. You can then use a Call Data block to attach the returned data to the current interaction.

If there are no matches, you can use the **Navigation** tab on the **Nothing Matched** block to go to a specified block, skipping the **Call Data** block.

**Match Criteria tab**

Use the settings on this tab to specify which criteria to compare to the original request. You must specify at least one item.

**Example**
Notice that Designer automatically creates child blocks for each outcome, which you can then use to specify the action to take if a match is found:

Routing Parameters

For Click-to-Call-In delayed scenarios, routing parameters are returned if there is a unique match. Use the settings on this tab to specify the variables that will hold the values for each parameter.

You can select variables for the following parameters:

- Outbound VQ
- Target Skill Expression
- Current Priority
Click-to-Call-In Match Block

- Priority Increment Interval
- Priority Increment
- Maximum Priority
- Notification Timestamp (when the push notification was initiated)

For more information, see Click-to-Call-In (Delayed).

Results tab

Specify the variable that will hold the retrieved storage information if a match is found.

**Example**

![Click-to-Call-In Match Block](image)
Go To Block

Contents

- 1 Using this Block
Go To Block

- Administrator

Use this block to jump to another block in the application.

**Related documentation:**

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.

**Properties - Go To**

This block is used to break the normal linear flow of the application, and jump directly to an out-of-order block. The target block should either be in the same phase, or can be the beginning of the Assisted Service or Finalize phases.

- By Name
- By Type
- By Description
- By Comment

Search then Choose target block to redirect to:
Return Block

Contents

• 1 Using this Block
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

Contents

• 1 Conditions tab
• 2 Milestone tab
This block enables an application to take a different path when certain conditions are met.

**Related documentation:**

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'
  ```

---

**Conditions tab**

Click **Add Condition**. The **Condition Expression** field is pre-populated with a sample expression that you can edit by typing a condition to evaluate.

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 also 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.

**Properties - Segmentation - decide how to route call**

This block is used to evaluate expressions and take different paths in the application based on the outcome. E.g. `varZipCode==94014` can be used to take a different path vs `varZipCode==95125`.

<table>
<thead>
<tr>
<th>Segment Label</th>
<th>Condition Expression</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Call</td>
<td><code>varServiceType == &quot;&quot;</code></td>
<td></td>
</tr>
<tr>
<td>Battery help</td>
<td><code>varServiceType == 'battery'</code></td>
<td></td>
</tr>
<tr>
<td>Electronic help</td>
<td><code>varServiceType == 'electronic'</code></td>
<td></td>
</tr>
<tr>
<td>All other cases</td>
<td><code>varServiceType == 'other'</code></td>
<td></td>
</tr>
</tbody>
</table>

**Milestone tab**

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

Contents

• 1 Shared Modules
• 2 Templates modules
• 3 External modules
  • 3.1 Input parameters
  • 3.2 Output parameters
  • 3.3 Usage and limitations
Shared Module Block

Use this block when you want to reference a Shared Module in an application.

**Related documentation:**

Shared Modules are useful for reusing code from multiple applications, as well as for splitting larger applications into smaller and more manageable segments. 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.

From the **Module** tab, you can select from the following types of modules:

- Shared Modules - Small applications managed from the Shared modules page.
- Templates - Used with Callback (read-only).

**Shared Modules**

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 an incompatibility with the latest version.

For more information about how to create and manage shared modules, see Shared Module.
On the **Signature** tab, you can then specify the values for the **Input** and **Output Parameters** to be used with the module you selected. You can use literals, variables, or expressions.

---

**Properties - SM: BEC Greeting Check**

This block can be used to invoke a shared module.

- **Module**
- **Signature**

- **Shared Modules**
- **Templates**

**Select a module:**

<table>
<thead>
<tr>
<th>Version</th>
<th>Label</th>
<th>Note</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latest</td>
<td>Use latest unpublished save.</td>
<td>01/22/2016</td>
</tr>
<tr>
<td>2</td>
<td>Version 2</td>
<td>Version 2</td>
<td>12/17/2015</td>
</tr>
<tr>
<td>1</td>
<td>Version 1</td>
<td>Version 1</td>
<td>12/17/2015</td>
</tr>
</tbody>
</table>
Tip

In general, you should select **Use latest unpublished save** rather than a specific version (unless there is a specific reason to do so, such as an incompatibility issue). This allows any applications that reference the module to always point to the latest version of the module. Otherwise, if you make changes to a version of the module and create a new version, you would also need to update and republish each application (or other module) that referenced the previous version.

Templates modules

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

Use this block to end the interaction and jump directly to the Finalize phase of the application.

**Related documentation:**

You can use the **Terminate Call** block to disconnect the customer and stop the interaction. Everything after the **Terminate Call** block is skipped, and the application moves directly to the Finalize phase.

### Properties - Terminate Call

This block is used to disconnect the call.

When added, processing will jump ahead to the beginning of Finalize phase and bypass any remaining blocks in the current phase.

- [ ] Stop the interaction immediately (digital only)
- [x] Finalize this interaction (digital only)

If the application is enabled for Digital (omni-channel) support, you can select **Stop the interaction immediately** to immediately terminate a digital interaction (such as a chat) and jump directly to the Finalize phase of the application. Otherwise, the interaction will be terminated after the blocks in the Finalize phase are processed.

**Finalize this interaction** applies to digital applications only and is enabled by default. If you disable this option, an Engage cloud Email or workitem interaction is not finalized or completed when it terminates. Instead, the interaction is sent back to the Universal Queue in Intelligent Workload Distribution (IWD).

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 interaction, such as Business Hours or Special Day, when the **Terminate Call** option is enabled.
Terminate Block (Digital)

Use this block to end the interaction and jump directly to the **Finalize** phase of the application.

**Related documentation:**

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 directly to the **Finalize** phase.

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

**Properties - Terminate**

- **This block is used to terminate the interaction.**

  When added, processing will jump ahead to the beginning of Finalize phase and bypass any remaining blocks in the current phase.

- **Stop the interaction immediately (digital only)**

- **Finalize this interaction (digital only)**

You can select **Stop the interaction immediately** if you want Designer to terminate the interaction immediately at this block and jump to the **Finalize** phase. Otherwise, the interaction will be terminated after the blocks in the **Finalize** phase are processed.

**Finalize this interaction** applies to digital applications only and is enabled by default. If you disable this option, an Engage cloud Email or workitem interaction is not finalized or completed when it terminates. Instead, the interaction is sent back to the Universal Queue in Intelligent Workload Distribution (IWD).
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.
Change Persona block

Contents

- 1 Using this block
Use this block when you want to dynamically change the persona being used by an application throughout the course of an interaction.

**Related documentation:**

Using this block

**Important**

This block is not available in Digital-type applications.

Switching personas can be useful when you want to use certain personas for specific situations. For example, you might want to use a more formal persona when dealing with sensitive customer issues, or switch to a different persona depending on the customer segment being served ("Gold" customers get one type of persona, "Blue" ones get another).

You can add this block to the Self Service or Assisted Service phases of the application, or use it in a Shared Module. After you've added the block, select the persona (or the variable) you want to use.

The selected persona will apply to any blocks that are using Text-to-Speech (TTS) services, such as Play Message, User Input, Menu, Bot, Record Utterance, Business Hours, Emergency, Special Day, Route Call, and Route Agent.
Change Persona block

For more information, see Personas.
User Interaction Blocks

Contents

• 1 Bot
• 2 Menu
• 3 Play Message
• 4 Record
• 5 Record Utterance
• 6 User Input
• 7 Chat Message
• 8 Chat Transcript
• 9 Get Chat Transcript
• 10 Send Email
These blocks help you to interact with customers, such as by offering menus, collecting information, or playing messages.

**Related documentation:**

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 type applications will see blocks related to email.

Use the links below to learn more about each block.

**Bot**

Add a chatbot resource to your application.

*Used in:* **Self Service**

**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
Sends a chat message to a contact.
Used in: **Assisted Service**

Chat Transcript
Emails the chat transcript to a contact.
Used in: **Assisted Service, Finalize**

Get Chat Transcript
Provides access to the latest chat transcript.
Used in: **Assisted Service, Finalize**

Send Email
Sends an email containing a standard-response message to a user.
Bot Block

Contents

• 1 About bots in Designer
  • 1.1 See how it works
• 2 Using this block
• 3 Intents and Slots tab
  • 3.1 Configure Bot details
  • 3.2 Configure intents
  • 3.3 Intents and Slots assignment
  • 3.4 Select a variable to send context to the Bot Session
  • 3.5 Invoking a Dialogflow ES bot with events
  • 3.6 Invoking a Dialogflow CX bot with events
• 4 Input Settings tab
• 5 Retry tab
  • 5.1 Use application-wide retry
  • 5.2 Allow retries
  • 5.3 Send No Input Event to Bot
• 6 Results tab
  • 6.1 Bot responses
  • 6.2 Bot status flags
  • 6.3 Additional bot information
  • 6.4 Viewing the results data
• 7 Adding logic for handling intents
Use the Bot block to add a chatbot to your application.

**Related documentation:**

- Administrator

About bots in Designer

Bots are software applications that leverage natural language processing and natural language understanding to recognize input and respond to customers in a way that resembles a conversation with a live agent. They can determine what a customer wants to do based on natural language input and then proceed with collecting the information required to fulfill the request or intent.

If you have a bot configured with a supported bot service provider, you can add it to the Designer Bot Registry. You can then use a Bot block to integrate the bot service with a Designer application.

The Bot block typically collects input from the customer, sends it to the external bot service, and waits for a response. This response may trigger completion of the Bot block (i.e. success or error) or it may trigger another turn of playing back a prompt to the customer and collecting additional input, which is again sent to the external bot service. These turns are handled internally in the Bot block and there is no need for the application developer to add more blocks to handle these.

See how it works

Watch this video to learn more about using bots in Designer applications.

**Link to video**

Using this block

The Bot block is located in the User interaction section of the palette. Add it to the Self Service phase of a Default application when you want to use a bot in your application. If the application is enabled for omni-channel, the same bot can service both voice and chat customers.

You can use multiple bots in an application. Simply add a Bot block for each bot you want to use.
Important
For voice calls, the Bot block plays questions and responses that are returned from the external bot via text-to-speech (TTS) prompts. With the exception of Dialogflow CX bots (which can be configured to use media sources that contain recorded audio), using pre-recorded audio prompts with the Bot block is not supported.

Intents and Slots tab
Use these settings to tell Designer about the bot resource you want to use in your application.

Configure Bot details
Specify the Bot provider and Name:

Configure Bot details

Bot provider
Dialogflow

Bot Name
DesignerDialogflowintegration

Once selected, Designer can automatically populate the block properties with intents and slots for the specified bot resource. Intent child blocks are hidden by default, but you can view these by clearing the Do not use intent child blocks checkbox:

Configure intents

Do not use intent child blocks

The intents that you enable (see Configure intents) are then added as intent child blocks.

Important
This option is not displayed if you are using a Dialogflow CX bot. Intent child blocks are not applicable to Dialogflow CX bots.
Configure intents

**Important**

For Google Dialogflow CX bots

- You can skip the settings in the Configure intents section as these are not applicable to Google Dialogflow CX bots.
- Intent child blocks (or any other type of nested or child blocks) are not supported for Dialogflow CX bots, even if they are shown by Designer. The Error Handler block, however, is supported and should be used to handle errors. All other exits from the Bot block will proceed directly to the next block.

An **intent** is something that the customer wants to accomplish, such as booking a trip or making a reservation. These are defined in the bot and the bot is set up to collect the information it needs to fulfill these intents, typically referred to as slots. **Slots** (also known as **entities**) provide additional context to the intent.

For example, let's say a bot detects that a customer wants to schedule an appointment. It now has the intent, but it also needs to know other details about the customer's request, such as the time, date, and the type of appointment. These are the slots, which the bot uses to determine the questions it needs to ask in order to collect the information needed to fulfill the customer's intent.

Select the intents you want to enable for the bot:

<table>
<thead>
<tr>
<th>Intent</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Fallback Intent</td>
<td>✔️</td>
</tr>
<tr>
<td>options</td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td></td>
</tr>
<tr>
<td>Default Welcome Intent</td>
<td>✔️</td>
</tr>
<tr>
<td>order_drink.different_card</td>
<td></td>
</tr>
<tr>
<td>Schedule Appointment</td>
<td>✔️</td>
</tr>
</tbody>
</table>

For each intent that you enable, Designer automatically creates a corresponding **Bot Option** child intent block in the application flow.
Once the external bot tells the Designer application it has identified an intent, Designer executes that specific intent's child block and any child blocks below that intent block (remember that intent child blocks are not supported for Dialogflow CX, even if they are shown by Designer). This works best for a small number of intents and is not recommended for bots that have more than 10 intents. Instead, for bots with larger numbers of intents, use a Segmentation block immediately following the Bot block to process specific intents and execute fulfillment (for more information, see Adding logic for handling intents).

Intents and Slots assignment

**Important**

*For Google Dialogflow CX bots*

- You can skip the settings in the **Intents and Slots assignment** section as these are not applicable to Google Dialogflow CX bots.

This tab allows you select variables to store values for the selected **Intent** and the related **Slots**.
It is recommended to do this only for bots that have a small number of intents and slots. For bots that have more than 10 intents, it is recommended to use the Results tab to capture all of the information returned from the bot and then use it in Assign blocks. This keeps the Bot block relatively isolated from the structure of the bot.

Select a variable to send context to the Bot Session

This option enables you to pass an initial slot (or entity) value to a Lex or, Dialogflow ES or CX bot. This can be useful when an attribute is known before the interaction starts, such as the customer's name, phone number, or email address. With this slot already filled, the bot won't need to prompt the customer to provide this information if the Bot block sends it to the external bot.

To use this option, you'll need to set up a variable that contains a JavaScript object that defines the value you want to pass to the bot. Then, select this variable from the dropdown. This also requires some configuration with your bot service provider, as you'll need to define an input context (or session attribute) for the slot and assign it a default value that corresponds to the JavaScript object. We've included an example that shows how to do this with a Dialogflow bot, but you can refer to the documentation from your bot service provider for additional information.

Example (Dialogflow ES)

For a quick example of how this works, let's set this up with a Dialogflow ES bot. First, we'll go to the **Intents** section of the bot and add a new **Context**. In this example, we've created an intent called **Intent with context** and added an input context to it, called **ExampleContext**.
Then, for the slot that we want to pass an initial value to, we need to set a default value for an attribute of the context. To do this, we'll go to the Action and parameters section and add the details for the slot we want to fill with an initial value. In this case, we'll add details for the phone number attribute.

To assign the default value, we'll hover on the right-side of the row to reveal the additional options menu and click it to open the Default value setting:

Now we can set the default value to match the name of the context and the attribute we want to fill:

<table>
<thead>
<tr>
<th>NAME</th>
<th>ENTITY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>phone-number</td>
<td>@sys.phone-number</td>
<td>$phone-number</td>
</tr>
</tbody>
</table>

In Designer, we'll create a user-defined variable called varInput. For its value, we'll add a JavaScript object called ExampleContext that passes an initial value of 1234567 to the phone-number attribute.
Tip
For Dialogflow ES bots, the JavaScript object must contain the context name. Lex bots, however, do not use contexts. JavaScript objects for a Lex bot can use any name, but if multiple contexts are passed to the bot, it only accepts the first one and ignores the others.

```javascript
{
  'ExampleContext':{
    'attributes': {
      'phone-number': '1234567'
    },
    'lifetime': 1
  }
}
```

Tip
Dialogflow CX bots support additional functionality using this method. See Invoking a Dialogflow CX bot with events.

In the Bot block, we can then select this variable as the context to pass to the bot:

Select a variable to send context to the Bot Session

```javascript
varInput
```

Another example of how you could use this option is to pass an initial message to the bot to start a chat conversation. In the JavaScript Object, add a field called content that contains the message you want to send (e.g. "I want to book a hotel room."): 

```javascript
{
  'ExampleContext':{
    'attributes': {
      'phone-number': '1234567'
    },
    'content': 'I want to book a hotel room.',
    'lifetime': 1
  }
}
```

Warning
The word `content` is a reserved keyword. Do not use `content` as the name of the variable that is passing context to the bot.

### Invoking a Dialogflow ES bot with events

You can use an **Event** to initiate a bot interaction with a Dialogflow ES bot without requiring the customer to provide any input. The context is still passed normally when invoking the bot with an event. To invoke the bot with an event, go to your Dialogflow bot settings and set the **Event** field of the context object to the name of the event you want to invoke. For example, we'll add an event called `sample-event` to an intent:

- **Intent with context**

```javascript
{
'ExampleContext':{
'attributes': {
'phone-number': '1234567'
}
},
'event': 'sample-event',
'lifetime': 1
}
```

If you set both an event and an initial message in the JavaScript Object, the bot ignores the initial message and uses the event.
Invoking a Dialogflow CX bot with events

You can also use an Event to initiate a bot interaction with a Dialogflow CX bot without requiring the customer to provide any input. The context is still passed normally when invoking the bot with an event. The functionality is similar to that of Dialogflow ES bots, but the structure of the JSON object has additional fields and certain naming conventions must also be observed (noted below).

Example:

```json
{
    'parameters': {
        'attributes': {
            'phone-number': '1234567'
        }
    },
    'content': 'I want to book a hotel.',
    'lifetime': 1
}
```

Context fields for Dialogflow CX bots

You can use the JSON object to pass the following properties to the Dialogflow CX bot:

<table>
<thead>
<tr>
<th>JSON Object Property</th>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>parameters</code> attributes object</td>
<td>Passes known context to the bot.</td>
<td>Each property of this object is sent to the external bot, which enables it to pre-fill certain slots with these values. <strong>Note:</strong> This property must be named <code>parameters</code>.</td>
</tr>
<tr>
<td><strong>event</strong></td>
<td>Invokes an intent directly in the external bot.</td>
<td>This skips the first input collection and allows the bot to process attributes that are passed to it directly without having to rely on customer input.</td>
</tr>
<tr>
<td><strong>content</strong></td>
<td>A string that contains an initial message.</td>
<td>This passes the string as the first input to the bot as if it was collected from the customer.</td>
</tr>
<tr>
<td><strong>webhookHeaders</strong></td>
<td>A JSON object that is passed from the CX bot to Designer and then passed back to the bot from Designer.</td>
<td>These values can be used by the bot fulfillment code to call an external API, query a database, etc.</td>
</tr>
<tr>
<td><strong>webhookPayload</strong></td>
<td>A JSON object that is passed from the CX bot to Designer and then passed back to the bot from Designer.</td>
<td>These values can be used by the bot fulfillment code to call an external API, query a database, etc.</td>
</tr>
</tbody>
</table>

Input Settings tab

Use this tab to configure settings and options related to the inputs customers provide to the bot service.
If you are setting up a Google Dialogflow bot, you can select **Use Streaming Audio** to have Designer stream the audio inputs directly to the bot services provider. (Note that Google Dialogflow CX bots use streaming audio only.)

If **Use Streaming Audio** is enabled, you can also select a barge-in option. Select **Disable barge-in** to prevent customers from interrupting the playback of bot prompts with voice or DTMF inputs. If you want the bot service to manage how barge-in is handled, you can select **Use barge-in settings from Bot** to override the Designer settings.

The **Beep before listening for input** option is available only if **Use Streaming Audio** is not enabled. When enabled, a tone is played after the bot asks the customer for input. The **Bot** block only recognizes the input that is received from the customer after the beep has played. You can then adjust the **Input timeout** values for both voice and chat inputs. These settings tell Designer how long to wait (in seconds) before assuming that the customer did not provide any input to the bot.

---

**Important**

**For Google Dialogflow CX bots**
Genesys recommends using the No speech timeout configured in Google Dialogflow CX and the corresponding no input handlers to control no inputs, instead of the Designer voice Wait for timeout and Retry prompting in the Bot block. To ensure the Google Dialogflow CX configuration is used, the Designer Bot block voice Wait for timeout should be set to a high value of at least the sum of the following:

- Longest possible prompt that might be played in a Google Dialogflow CX turn
- Google Dialogflow CX No speech timeout duration
- 3s buffer

If you are using a Google Dialogflow CX bot, you can also manage the following options (these are not available for other bot types):

- Enable Sentiment Analysis
  You can enable this option if the Dialogflow CX bot is performing sentiment analysis while detecting intents. The bot service analyzes the input to determine the overall attitude of the customer (e.g. positive, negative, or neutral) and returns the result to Designer in a variable.

- Send DTMF input to the bot
  If your Dialogflow CX bot is configured to accept DTMF inputs, enabling this option allows both voice and DTMF inputs to be sent to the bot. You can then configure the Input termination character, Interdigit Timeout, and Terminating Timeout settings (see DTMF settings for more information about these settings). When this option is enabled, customers can provide DTMF input at any point in the conversation where they would typically provide voice input. If both voice and DTMF input are provided simultaneously, the DTMF input is given priority and the voice input is ignored.

  **Important**
  No Input timeout settings apply to both voice and DTMF inputs. After the first DTMF input is provided, the Input termination character and Interdigit Timeout settings are used to determine when the DTMF input has ended.

- Use Inband DTMF is available when the Send DTMF input to the bot option is enabled. When this is enabled, DTMF settings configured in the Google Dialogflow CX bot are respected.

  The Confidence Level slider can be used to adjust the confidence level threshold for the Bot block. This property is not sent to the bot provider; it is used only by Designer to determine if a response from the bot is a No Match. If the confidence received from the bot is less than the specified threshold level, then Designer treats it as a No Match. Otherwise, the response is handled normally.

  **Important**
  If the Confidence Level slider is set to 0 (zero), Designer never generates a No Match for the Bot block.
Retry handling in the Bot block works a bit differently than it does in other blocks with retry settings, such as Menu or User Input. Bot services typically have their own retry behavior already built-in. For example, the bot automatically asks the customer to repeat a response that it didn't understand. Rather than handling retries based on a single question and a single response (as in Menu or User Input blocks), the Retry settings on the Bot block are based on the conversation taking place between the bot and customer. This conversation can consist of several questions and responses, all encapsulated within one Bot block.

You can use the settings on this tab to set up the Bot block to control retry handling (see below) or trigger an event that allows the external bot service to control all retry handling (see Send No Input Event to Bot).

Use application-wide retry

Select this option if you want to use the retry settings that are specified on the Global Retry tab in the Application Settings.

Allow retries

Select this option to specify retry rules for this block. When enabled, you can set the following options:

Number of No Input retries allowed

Select the number of retries to allow for each question and response sequence that occurs in the conversation between the bot service and the customer.

Specify retry prompt and destination if the user's input isn't recognized

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 customer.

### No Input #1

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td></td>
<td>No input one</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

**Number of No Match retries allowed**

Most bots will follow-up with the customer if they don't understand the input that's been provided. For example, the bot will simply ask the customer to repeat the information until it successfully captures the response. As this type of handling is typically built-into the bot by the bot services provider, you may not need to specify this setting in the **Bot** block.

**After Final No Input**

Add the prompt to play after the maximum number of permitted **No Input** retries is reached. As 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. As 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td></td>
<td>No match final</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

**Send No Input Event to Bot**

*(This option is supported for **Google Dialogflow ES** bots only.)*

If the external bot service is handling retry behavior, you can select this option to send a **No Input** event to the bot service.
In this scenario, all retry handling is performed by the external bot service. The Bot block sends the specified Event Name to the external bot, which then customizes the retry behavior for each conversational turn.

**Important**

If the external bot service is handling all retries, the No Input and No Match totals are not tracked in the Session Detail Records (SDR).

Results tab

Specify the variables that will hold various results data, as returned to the Bot block from the bot services provider. Each variable is described in more detail below.
Bot responses

Store latest response from bot

This variable stores details about the latest conversation that the bot engine had with a customer. For example, the results for a Dialogflow bot used to book a ghost removal service might look like this (JSON formatted):

```json
{
    "status": {
        "code": 0,
        "message": null
    },
    "data": {
        "botName": "MySampleServiceBookingBot",
        "botAlias": null,
        "sessionId": "ABC123",
        "state": "SUCCESS",
        "intent": "Book a Ghostbuster",
        "intentScore": 1,
        "slots": {
            "neighbourhoods": "Queens",
            "location": "backyard",
            "date": "2020-01-11T12:00:00-05:00",
            "ghost": "Zuul the Gatekeeper"
        },
        "slotsData": null,
```
In the example above, some of the details that were returned include:

- **botName** – name of the bot that was invoked.
- **sessionID** – unique ID assigned to the session.
- **state** – indicates SUCCESS if everything worked.
- **intent** – the intent that was detected (i.e. what the customer wanted to do).
- **slots** – the details that the bot collected from the customer to fill the associated slots (or entities) for the intent.
- **inputTranscript** – the utterance (voice or chat input) that the bot received from the customer.

### Store bot invocation result code

This variable stores the HTTP status code received from the bot when it was last invoked by the application. For example, a result code of 200 (OK) indicates that the bot was successfully invoked. Other result codes, such as 401 (Unauthorized) or 403 (Forbidden), can indicate there was a problem reaching the bot service.

### Bot status flags

#### Bot invocation or system error

If true, this indicates that Designer was able to successfully invoke the bot service without any errors. If false, an error was encountered and the bot was not successfully invoked. There might be an issue with the system, bot service, or you might need to check the credentials provided for your bot service in the Bot Registry.

#### Bot engine execution error

If this value is undefined, this indicates that no errors occurred. If true, this indicates that Designer was able to communicate with the bot, but an error occurred while the bot engine was processing the request. For example, the bot returned an incorrect response and triggered the Error Handler block. If it returns false, it means that there was an error with the bot or the system.

### Additional bot information

The variables in this section are applicable only to Dialogflow CX type bots. Dialogflow CX bots manage conversations differently than other types of bots. If you are using a Dialogflow CX bot, you can set these variables to capture additional details about the interaction.

These properties may be supported for other bot types in future versions of Designer.
Store all slots from the bot

This variable stores an object (JSON formatted) that contains all of the slots that were returned from the bot provider at the end of the last turn with the external bot. It does not contain intents. For example:

```json
{
    "operation": "Purchase Item",
    "order_unit": "phone",
    "location": {
        "original": "main st",
        "street-address": "main st"
    },
    "operation_complete": false,
    "admin-area": null,
    "is_returning": true,
    "zip-code": null,
    "city": null,
    "isLogged_in": false,
    "cart": "Nest Thermostat, phone",
    "has_welcomed_user": true
}
```

Capture slots information with every intent

This variable captures slots organized by intents as these intents were identified in chronological order. Each intent captures slots as they were at the time the intent was identified. This gives an additional perspective into the conversation with the bot. For example:

```json
[
    {
        "intent": "Default Welcome Intent",
        "slots": {
            "has_welcomed_user": true
        }
    },
    {
        "intent": "retail.purchase_item_initiate",
        "slots": {
            "operation_complete": false,
            "operation": "Purchase Item",
            "has_welcomed_user": true,
            "is_returning": true
        }
    },
    {
        "intent": "small_talk.confirmation.proceed_as_guest",
        "slots": {
            "has_welcomed_user": true,
            "operation_complete": false,
            "is_returning": true,
            "isLogged_in": false,
            "operation": "Purchase Item"
        }
    }
]
```
Store the reason the interaction ended

If the bot conversation is successful (it does not error out), this variable stores information about how the bot session ended. There are two possible result values:

- **END_INTERACTION** – the session ended normally.
- **LIVE_AGENT_HANDOFF** – the customer should be transferred to a live agent and the application should try to move to Assisted Service. This must be handled by the application logic. There is no in-built behavior that would start Assisted service once this result is returned by the external bot.

If the bot does not return a SUCCESS state (for example, the bot experienced an error), this variable is not updated.

Viewing the results data

You can view the results data in Designer Analytics by going to the Session Detail Records dashboard. In the **All Events** panel, find the application instance you want to check and then filter or search for the data you want to view.

For example:

```
variables.varBotInvocationResultCode
variables.varBotIsExecutionError
variables.varBotIsSystemError
variables.varBotLatestResponse
```

Adding logic for handling intents

To specify additional logic for handling intents, Genesys recommends using a Segmentation block to define different pathways for the application to take when certain intents are detected.

In this example, a segmentation block is configured as **Intent Fulfillments**, with conditions added based on intents:
For each intent added as a condition, Designer creates a corresponding **Segment** block in the application flow. You can then build additional logic for an intent by placing child blocks under these **Segment** blocks. For example, you might want to call a Shared Module that fulfills that intent.

For more information about setting up segmentation blocks and condition expressions, see the Segmentation block page.
Menu Block

Contents

• 1 DTMF Options tab
• 2 Menu Prompts tab
  • 2.1 Input timeout
  • 2.2 Disable barge-in
• 3 Retry Prompt tab
  • 3.1 Allow Retries
• 4 Results tab
• 5 Milestone tab
Use the Menu block to present a list of choices to customers.

**Related documentation:**

You can use the **Menu** block only in the **Self Service** phase to present a list of choices to the customer and accept a selection that the customer provides.

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.

**Warning**

The **Menu** block is not supported with **SMS** media.

**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 voice or chat input was received from the customer.

Disable barge-in

Select this option to prevent customers from interrupting a prompt while it is still playing. For example, you might want a "Welcome" message to play all the way through before they 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 customer.
Important
The selected barge-in setting applies irrespective of whether global DTMF options 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.

You can also specify prompts to play for each enabled DTMF option. For chat applications, the chat widget displays the **DTMF Key** prompts as *quick reply* buttons. The customer can then select one of the quick replies instead of manually typing a response.
Retry Prompt tab

Allow Retries

Select to allow customers 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 customers 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 customer.

- **Number of No Match retries allowed**
  Enter the number of retries to allow for customers 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 customer.

- **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

Contents

• 1 Call Handling tab
  • 1.1 Terminate the call
• 2 Play Audio tab
  • 2.1 Disable barge-in
  • 2.2 Always play prompt and disable buffering
• 3 Navigation tab
• 4 Set Variables tab
• 5 Milestone tab
Menu Option Block

- Administrator

This block appears automatically in the Application Flow if at least one DTMF key is enabled in a Menu block. You can use it to set up handling for when the associated option is selected.

**Related documentation:**

- 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 interaction if this menu option is selected by the user.

Optionally, you can choose to route this interaction if this menu option is selected by the user. If so, select a Skill and Virtual Queue to which the interaction 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 interaction 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 users 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 user 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 user.
Important
The selected barge-in setting applies irrespective of whether global DTMF options are used or not.

Always play prompt and disable buffering

Select this option if you want users 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 for a voice call and the user 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 user.

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

Properties - Main - Sales

Menu Option blocks can be used to specify common operations if the DTMF key associated with this option is pressed.

<table>
<thead>
<tr>
<th>Option key</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify block label</td>
<td>Main - Sales</td>
</tr>
</tbody>
</table>

Specify actions in tabs below if this Menu Option is selected. All these actions are optional.

- Call Handling
- Play Audio
- Navigation
- Set Variables
- Milestone

Select options to enable going back to a previous menu if this Menu Option is selected.

- Go to previous menu (played before this Menu block)
- Go to first level menu in the Self Service Phase
- Go to another block

| Assisted Service |

- Continue with normal processing. Do not go back to previous Menu blocks.

Set Variables tab

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

Contents

• 1 Using this Block
  • 1.1 Prompts tab
  • 1.2 Adding a new prompt
  • 1.3 Message Settings tab
• 2 Scenarios
Administrators

This block enables you to play a message to a customer. For example, you might play a greeting or provide some instructions.

**Related documentation:**

You can use the **Play Message** block in the **Self Service** and **Assisted Service** phases to play audio messages to the customer. 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 Media Resources page, or variables played as TTS.

### Using this Block

**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 options 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.

Adding a new prompt

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).

- **Record Utterance** — Play the recorded file captured by the Record Utterance block. (This option is only supported in the Self Service phase, as the recording file captured by the Record Utterance block is no longer available after the Self Service phase.)
- **External Audio** — Play an audio file hosted outside of Designer. For example, you could define a variable that points to an audio file hosted on another server, and specify that variable here.

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. If the prompt is an **Announcement** type, you can specify the following settings to specify the duration of the announcement (for example, to interrupt the hold music at specific intervals to play an announcement that provides the caller with their estimated wait time):

- **Audio length** (milliseconds) - Specifies how much of the audio file to play. The default audio length is the actual length of the referenced audio.
- **Time offset** (milliseconds) - Specifies the time offset of when to begin playing the referenced audio. The default time offset is 0 (ms).

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[ap]??, 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).
Message Settings tab

Rich media settings

**Image URL**

(Optional) Enable this option if you want to display an image to customers during chats. Specify the variable that contains the URL to the image that you want to display.

**Add Field Code**

(Optional) If you are using field codes, you can specify them here.

**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
Play Message Block

- **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

Contents

• 1 Things you should know
• 2 Recording tab
• 3 Advanced tab
  • 3.1 FCR
  • 3.2 GIR
• 4 Results tab
• 5 Updating applications to use a new recording type
  • 5.1 Moving from FCR to GIR
  • 5.2 Moving from GIR to FCR
This block enables you to control call recording. For example, you can start or stop recording when certain conditions are met.

**Related documentation:**

- Administrator

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

Contents

- 1 Prompts tab
- 2 Advanced tab
  - 2.1 Maximum recording duration
  - 2.2 Minimum recording duration
  - 2.3 End of recording timeout
  - 2.4 Use any DTMF keypress to stop recording
- 3 Retry tab
- 4 Results tab
Record Utterance Block

• Administrator

This block enables you to capture a voice recording of a customer.

**Related documentation:**

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

Use the settings on this tab to specify the recording duration and timeout settings.

Maximum recording duration

Specify the maximum duration (in seconds) of the recording. The default maximum recording duration value is 10 seconds.

Minimum recording duration

Specify the minimum duration (in milliseconds) of the recording. The default minimum recording duration value is 250 ms (milliseconds).

End of recording timeout

Specify how many seconds of silence should elapse before recording stops. The default recording timeout value is 2 seconds.
Use any DTMF keypress to stop recording

Enabling this option allows users to stop the recording with any key press.

---

**Advanced Settings**

- **Maximum recording duration**: 30 seconds
- **Minimum recording duration**: 250 ms
- **End of recording timeout**: 2 seconds
- **Use any DTMF keypress to stop recording**

---

**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

Contents

• 1 Prompts tab
  • 1.1 Disable barge-in
• 2 Input Tab
  • 2.1 Built-in Grammar
  • 2.2 External Grammars
• 3 Confirmation tab
  • 3.1 Never
  • 3.2 Always
  • 3.3 Within a specified confidence range
  • 3.4 Specify prompts...
• 4 ASR Settings tab
  • 4.1 Use application-wide ASR settings
• 5 DTMF Settings tab
• 6 Retry tab
  • 6.1 Use application-wide retry
  • 6.2 Allow retries
• 7 Confirmation Retry tab
  • 7.1 Use default confirmation strategy
  • 7.2 Allow retries
• 8 Results tab
• 9 Milestone tab
• 10 Example scenario
• 11 Dynamic Grammars
  • 11.1 Using an **HTTP REST** Block
  • 11.2 Using an **Assign Variables** Block
User Input Block

Administrators

This block enables you to collect information from the customer.

**Related documentation:**

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.

**Example**

![Properties - User Input](image)

This block is used to ask a question and collect input from the user. It provides options for multiple attempts.

**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
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.
- **Service?** - If selected, Var? is enabled and you can select the variable for the desired service. **Important:** A grammar can either be Dynamic or a Service; it cannot be both at the same time.
- **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.
- **Arg** - If Service is selected, you must specify the variable to be matched against the query string contained within the Service variable. **Important:** If you don’t specify an Arg variable, you’ll get a validation error prompting you to include an argument in the Service grammar.

**Confirmation tab**

Use the settings on this tab to enable and configure user input confirmation. This allows callers to review and confirm any spoken or DTMF inputs they have provided.

- **Never**
- **Disable user input confirmation.**
- **Always**
- **Enable user input confirmation.**
Within a specified confidence range

Enable user input confirmation for callers that meet the defined confidence threshold. If selected, you can then define the lower and upper thresholds.

Specify prompts...

If user input confirmation is enabled, specify the prompt(s) to play. You can also enable or disable barge-in (see the Prompts tab) for more information about this setting.

Example

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 determins 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.

## Confirmation Retry tab

Enable and configure the settings for user input confirmation retries.
Use default confirmation strategy
Select this option to use the default settings.

Allow retries
Enable retries. Specify the number of retries that are allowed for No Match, No Input, and Disconfirmations.

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

<table>
<thead>
<tr>
<th>Properties - User Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>This block is used to ask a question and collect input from the user. It provides options for multiple attempts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store output result (either DTMF entered digits, or the ASR utterance) in this variable (required)</td>
<td></td>
</tr>
<tr>
<td>varSurveyResponse</td>
<td></td>
</tr>
<tr>
<td>Store semantic interpretation in this variable</td>
<td></td>
</tr>
<tr>
<td>– choose variable –</td>
<td></td>
</tr>
<tr>
<td>Store confidence score in this variable</td>
<td></td>
</tr>
<tr>
<td>– choose variable –</td>
<td></td>
</tr>
<tr>
<td>Store the output result details in this variable</td>
<td></td>
</tr>
<tr>
<td>– choose variable –</td>
<td></td>
</tr>
</tbody>
</table>

The format of the output result details variable will be an object with the contents:

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>boolean</td>
<td>True if input was collected successfully.</td>
</tr>
<tr>
<td>interpretation</td>
<td>string</td>
<td>Interpreted value for the user input.</td>
</tr>
<tr>
<td>input</td>
<td>string</td>
<td>DTMF or 'voice'</td>
</tr>
</tbody>
</table>

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:

```json
{
    "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.
Automated Message Block

Contents

• 1 Using this block
• 2 Message tab
  • 2.1 Specify From Address
  • 2.2 Additional To and Cc Addresses
  • 2.3 Select a Message
• 3 Message Settings tab
• 4 Results tab
Use the **Automated Message** block to create and send an email to a customer.

**Related documentation:**

**Using this block**

You can use the **Automated Message** block to create and send an email to a customer. For example, when a customer sends an email to your contact center, you could use this block to send the customer a confirmation message that their email was received.

A few things to note:

- This block can only be used in a **Digital** type application to send an **email** message.
- The application must be associated with a mail box or an Open Media endpoint.
- Before you can select a message to use with this block, you must associate a **Digital Resources** collection with this application in the Application Settings.

**Message tab**
Specify From Address

Select the variable that contains the email address to be displayed as the From address.

Additional To and Cc Addresses

This setting is optional. You can choose to specify any additional To and Cc recipient email addresses, using variables or comma-separated lists.

Select a Message

Use the picker to browse the Digital Resources collection that is associated with this application and select the standard message you want to use.

Important

Custom messages are not supported. You must select one of the standard response messages from the Digital Resources collection.
Automated Message Block

Message Settings tab

Properties - Automated Message

This block is used to create an email.

Provide values to substitute to message field codes.

(Optional) If you are using field codes, you can specify them here.

Results tab
Specify the variable that will store the results of the email operation.
Chat Message Block

Contents

• 1 Messages tab
• 2 Field Codes tab
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

Contents

- 1 Custom Messages tab
Chat Custom Message Block

• Administrator

Use this block to send a custom chat message to a contact.

Related documentation:

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 cannot be used as a busy treatment. This block is only supported for Designer deployments that are using legacy chat services (i.e. Chat Version 8.x). If you are not sure if this applies to your deployment, contact your Genesys representative.

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 Custom Message Block

**Properties - Chat Custom Message**

This block is used to send a custom message to the caller.

**Custom Messages**

Nick Name

Specify custom messages to be sent

+ Add Message

<table>
<thead>
<tr>
<th>Var?</th>
<th>Value</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Thank you for waiting. A representative will be with you shortly.</td>
<td></td>
</tr>
</tbody>
</table>
Send Chat Transcript Block

Contents

- 1 Message Transcript tab
- 2 Message Settings tab
- 3 Field Codes tab
- 4 Results tab
Send Chat Transcript Block

• Administrator

Use this block to send a transcript of the chat to the email address specified in a contact's profile.

Related documentation:

The **Send Chat Transcript** block lets you send a transcript of the chat to the email address specified in the 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.

(Optional) If you are sending the transcript to multiple recipients, you can select the variables containing a comma-separated list of email addresses to be added as additional recipients or copied on the email.

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.
Send Chat Transcript Block

Properties - Send Chat Transcript

This block is used to send a copy of the chat transcript to the customer.

Message Transcript

Specify From address

Additional (optional) To addresses (comma separated list)

Additional (optional) Cc addresses (comma separated list)

Select a Message

Message Settings tab

Use this tab to specify substitute values for field codes.
Important

Avoid using single quotes (‘) in field values. If the value requires a single quote, use a backslash (\) to escape the quote character. For example, to add the name O’Malley as shown in the above image, you would enter 0\’Malley.

Field Codes tab

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

Results tab

Select the variable in which to store the results of the operation.
Get Chat Transcript Block

Contents

• 1 Using this block
This 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.

Related documentation:

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.

---

Properties - Get Chat Transcript

This block is used to get the latest version of the chat transcript.

Store the chat transcript in this variable

```varChatTran
```

The format of the chat transcript will be an ECMAScript Object (array), that contains transcript messages as elements. Each element has the following properties:

- `date`: number of seconds since 1 January 1970 00:00:00 UTC
- `device`: name of chat party
- `text`: chat message
- `visibility`: specifies the visibility level of this particular transcript event (could be: "ALL" - like conference mode, "INT" - like coaching mode, "VIP" - like monitoring mode for supervisors)

E.g. `[ { "date": 1510304070, "device": "system", "text": "Hi, welcome to FitBizz. A coach will be with you shortly;", "visibility": "ALL" }, { "date": 1510304074, "device": "system", "text": "Your estimated waiting time is 1 minutes;", "visibility": "ALL" } ]`

Store the outcome of the Get Transcript block in this variable

```varChatTranResult
```

The format of the outcome variable will be an object with the contents:

- `<var>success` - true | false
- `<var>error = 'error description'` (optional property)
Send Email Block

Contents

• 1 Using this block
• 2 Message tab settings
• 3 Results tab settings
Send Email Block

• Administrator

This block enables Designer to send an email that an agent created in advance.

Related documentation:

Using this block

The Send Email block is used in Digital type applications to send an email to a customer.

A few things to note:

• This block only sends an email that was created by an agent in advance. You cannot use this block to compose a message or specify recipients. (For creating and sending new emails to customers, see the Automated Message block.)

• You can specify an optional From address, if you want the email to display a different From address than the one used in the agent's email.

When Designer processes this block, a new application session is created to process and send the email. If the queue holding the agent's email is associated with a Designer application, the email is sent when that Designer application runs.

Message tab settings
This setting is optional. If you want to display a different From address that the one used in the agent's email, use this setting to select the variable containing the From address to be used.

Results tab settings

Store the outcome of the Email operation in this variable

varEmailResult

The format of the outcome variable will be an object with the contents:

- `<var>.success = true | false`
- `<var>.error = 'Timeout' (optional property)`
Specify the variable in which to store the results of the send email operation.
Send SMS block

Contents

• 1 SMS tab
• 2 Results tab
Send SMS block

The **Send SMS** block lets you send an SMS (Short Message Service) message to a customer. To use this block, your site must be enabled for Digital Channels. For more information, contact your Genesys representative.

**Related documentation:**

-  

SMS tab

Use the settings on this tab to specify the variables that hold the values for the customer’s phone number (i.e. ANI) and the call display number, or caller ID (i.e. DNIS).

Click **Add Message** to specify the SMS to be sent to the customer (or the variable that contains the message).
Send SMS block

Properties - Send SMS

This block is used to send an SMS to the caller.

SMS  Results

Select the variable that contains the customer phone number, to send the SMS to

ANI

Select the variable that contains the call display number (caller ID)

DNIS

Specify SMS to be sent

+ Add Message

<table>
<thead>
<tr>
<th>Var?</th>
<th>Value</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Your verification code is: ABCD1234</td>
<td></td>
</tr>
</tbody>
</table>

Results tab

Use the settings on this tab to specify the variables that will hold the results of the Send SMS block.
Send SMS block

---

**Properties - Send SMS**

This block is used to send an SMS to the caller.

- **sms**
- **Results**

This variable will be set to true if the sms block was successfully executed, and false otherwise

```
smsResult
```

This variable will contain the error (if any, in case sms message block encountered an error)

```
smsError
```
Business Controls Blocks

Contents

• 1 Business Hours
• 2 Data Tables
• 3 Emergency
• 4 Special Day
The Business Controls blocks control the various operational aspects of your business.

**Related documentation:**

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

Contents

• 1 Business Hours tab
  • 1.1 Open All Day
  • 1.2 No End Time
  • 1.3 Terminate the call if it is outside Business Hours
  • 1.4 Use Business Hours defined in Business Controls
  • 1.5 Timezone Override
• 2 Closed Messages tab
• 3 Results tab
• 4 Using Business Hour Values in Prompts
• 5 Scenarios
  • 5.1 Specify that the business is closed after 4 PM on Thursdays
  • 5.2 Play a message if a customer calls after business hours, and then end the call
  • 5.3 Specify business hours for a different timezone than the one the application is running in
  • 5.4 Specify business hours where the Start Time is later than the End Time
Use the Business Hours block to inform customers that your business is closed.

**Related documentation:**

You can use the **Business Hours** block in the **Initialize, Self Service**, or **Assisted Service** phase to inform customers that your business is closed. You can also choose to end the interaction 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 checks 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**

<table>
<thead>
<tr>
<th>Day</th>
<th>Start Time</th>
<th>End Time</th>
<th>No End Time</th>
<th>Open All Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>Open</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>9:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>9:00 AM</td>
<td>5:00 PM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Timezone Override**

-- choose variable --

If specified, business hours will be calculated using this timezone value, instead of the Application Timezone.
Example 2

Properties - Check Business Hours

This block checks the current time to see if it lies within closed hours. Closed hours are defined in this block itself. Messages can be set up to play if the caller encounters closed hours.

- **Business Hours**
- **Closed Messages**
- **Results**

- **Disable Terminate the call if it is outside Business Hours.**
- **Use Business Hours defined in Business Controls**

```plaintext
varReqBusHrs ▼
```

The variable must contain the name of one of the Business Hours defined in Business Controls.

- **Timezone Override**

  ```plaintext
  -- choose variable -- ▼
  ```

  If specified, business hours will be calculated using this timezone value, instead of the Application Timezone.

Closed Messages tab

In the **Closed Messages** tab, you can select the messages to play or send to customers when your business is closed. If your application is a Default application type, you can specify the prompts to play for voice or chat customers. For Digital application types, you can select an acknowledgement message from the Digital Resources Collection that will be emailed to customers.

**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 the application. If specified, the variable is assigned the Boolean value of `true` or `false`.

You can assign a variable to the **Store the number of minutes before the business is opened** to track how long until the business is open, if it is currently closed. If the business is open, this variable is set to 0. The maximum value returned is the number of minutes until the end of week (Sunday to Saturday). (This feature is supported only for Business Hours defined in Business Controls.)

You can assign a variable to the **Store the number of minutes before the business is closed** to track how long until the business is closed, if it is currently open. If the business is closed, this variable is set to 0. (This feature is supported only for Business Hours defined in Business Controls.)

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:

```json
{
  "days": [ 
    {"name": "Sunday","closed": true,"range": [{"starttime": ","endtime": "]"}],
    {"name": "Monday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Tuesday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Wednesday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Thursday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Friday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Saturday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
    {"name": "Sunday","closed": false,"range": [{"starttime": "1000","endtime": "1800"}]
  ]
}
```
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 PM 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 **Initialize** 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.

Specify business hours where the **Start Time** is later than the **End Time**

If you try to specify these types of business hours in the usual way, Designer displays an error ("Start time should be earlier than end time"). We'll use a business that is open from 6:00 PM to 9:00 AM as an example of how you can set this up.

• Create two sets of business hours. For the first set of business hours (primary), set the **Start Time** as 6:00 PM for the days you are open (e.g. Monday to Friday). For each open day, select the **No End Time** check box, as we've done here for our primary business hours called **JC_BH_1**:

<table>
<thead>
<tr>
<th>Day</th>
<th>Start Time</th>
<th>End Time</th>
<th>No End Time</th>
<th>Open All Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td>6:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>6:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>6:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>6:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>6:00 AM</td>
<td>Open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• For the secondary set, specify 12:00 AM as the **Start Time** and 9:00 AM as the **End Time** for Tuesday through Saturday, as we've done here for our secondary set of business hours called **JC_BH_2**:
Set up your application to first check the primary set of business hours. If the business is closed, have it then check the secondary set. If both sets of business hours are closed, the business is considered to be closed. Otherwise, the business is considered to be open between the desired hours of 6:00 PM to 9:00 AM.

<table>
<thead>
<tr>
<th>Day</th>
<th>Start Time</th>
<th>End Time</th>
<th>No End Time</th>
<th>Open All Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>12:00 AM</td>
<td>9:00 AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>12:00 AM</td>
<td>9:00 AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>12:00 AM</td>
<td>9:00 AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>12:00 AM</td>
<td>9:00 AM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>12:00 AM</td>
<td>9:00 AM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Tables Block

Contents

- 1 Data Table tab
- 2 Results tab
Use this block when you want an application to reference one of your data tables.

**Related documentation:**

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 can 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.

- **Strict comparisons** - Select this option to use a stricter form of key matching when performing lookups. If enabled, Designer won't trim or ignore leading zeros in numeric columns or spaces in string columns. This option also prevents type-coercion during lookups; for example, strings won't be able to match numeric keys.

- **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

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 --

Table

The count of returned rows will be stored in this variable.

-- choose variable --
Emergency Block

Contents

• 1 Using this Block
• 2 Scenarios
Use this block to add a conditional emergency option to your application.

**Related documentation:**

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 or send an emergency message and then optionally terminate the interaction. 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 interactions 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 interactions 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.

For Default application types, click **Add Emergency Prompt** to add one or more emergency prompts to play to callers when the emergency mode switch is set to **ON**. For Digital application types, select the message(s) to be emailed to customers.
Enable the **Terminate the interaction after playing emergency messages listed below** check box if you want the **Emergency** block to end the interaction 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.

**Properties - Emergency check**

- **This block is used only if its emergency mode is switched on. It can be used to enabled emergency mode in the Self Service phase and optionally jump to the Finalize phase.**

- **Use Emergency Flags defined in Business Controls**

  - **OFF**

  - **Emergency mode is OFF**

  - **When ON, this block will play any defined prompts and optionally terminate the call. If it is OFF, this block does not play messages or perform any other actions.**

- **Terminate the call after playing emergency messages listed below**

- **Store the result of the emergency flag check in this variable:**

  - **– choose variable –**

  If emergency flag is on, this variable is set to true.

**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 interaction after playing emergency messages listed below** check box.
  • Specify a variable to store the result of the emergency flag check.
Special Day Block

Contents

• 1 Holiday tab
• 2 Default Prompts/Messages tab
• 3 Results tab
• 4 Scenarios
Special Day Block

• Administrator

Use this block to define holidays and other special days.

**Related documentation:**

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 or send messages to announce closures or greetings. It can also terminate the interaction if your business is closed.

**Holiday tab**

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

Next, click on 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.
- For Default application types, you can enable **Play prompt for this holiday** to play a special greeting to customers during a special day.
- For Digital applications type, you can enable **Send message for this holiday** to email a special message to customers during a special day.
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

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

To

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>30</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>TTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Sorry we are closed for Christmas.
Default Prompts/Messages tab

Click Add Prompt to specify a prompt (or message, if the application is a Digital type) to play or send if the application receives an interaction on a special day, and that particular day does not have a custom prompt or message.

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

Contents

• 1 Query VQs
• 2 Route Agent
• 3 Route Call
• 4 Start Treatment
• 5 Voice Mail
• 6 Transfer
• 7 Route Digital
• 8 Predictive Routing
Use the Routing blocks to specify how an interaction should be routed when certain conditions are met.

**Related documentation:**

- Administrator

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 customers while their interaction 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 applications only) Routes a multimedia interaction to a target.
Used in: Assisted Service

Predictive Routing
Routes to the agent best equipped to handle the request.
Used in: Assisted Service
Query VQs Block

Contents

- Using this Block
This block enables your application to determine the Estimated Wait Time (EWT) for multiple virtual queues.

**Related documentation:**

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.
Query VQs Block

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

<table>
<thead>
<tr>
<th>Virtual Queue</th>
<th>Variable</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bronze</td>
<td>ewt1</td>
<td></td>
</tr>
<tr>
<td>Silver</td>
<td>ewt2</td>
<td></td>
</tr>
<tr>
<td>FD_Billing_Gold</td>
<td>ewt3</td>
<td></td>
</tr>
</tbody>
</table>

Estimated Wait Time will be saved in: Minutes

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

vc_best_ewt

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

ewt_best
Route Agent Block

Contents

• 1 Agent Routing tab
  • 1.1 Agent Routing
  • 1.2 Other Routing Settings
• 2 Treatments tab
  • 2.1 Audio
  • 2.2 Shared Module
• 3 Advanced tab
  • 3.1 Targeting
  • 3.2 Extensions
• 4 Results tab
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.

**Important**

The **Route Agent** block can only be used to route voice calls. It does not support the routing of chat interactions.

### 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.

**Important**
If you select the **Last Called Agent** option, you must also select the **Enable Customer Profile** option in the Contextual Data tab of the Application Settings.

**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.

   **Properties - Initialize**

   ![Initialize Properties](image1)

   This block or phase is typically used to setup variables for the application and initialize them. Assign blocks can be used to calculate expressions and assign their results to variables in this phase.

   **User Variables**

   ![User Variables](image2)

   **Specify User Variables. String values must be surrounded by single quotes.**

   ![Add Variable](image3)

<table>
<thead>
<tr>
<th>Name</th>
<th>In</th>
<th>Out</th>
<th>Default Value</th>
<th>Private</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>ewt</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

   **Properties - Segmentation**

   ![Segmentation Properties](image4)

   This block is used to evaluate expressions and take different paths in the application based on the outcome. E.g varZipCode==94014 can be used to take a different path vs varZipCode==95125.

   **Conditions**

   ![Add Condition](image5)

<table>
<thead>
<tr>
<th>Segment Label</th>
<th>Condition Expression</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWT less than or equal to</td>
<td>EstimatedWaitTime &lt;= ewt</td>
<td></td>
</tr>
<tr>
<td>EWT greater than last wai</td>
<td>EstimatedWaitTime &gt; ewt</td>
<td></td>
</tr>
</tbody>
</table>

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:

**Properties - Play Message**

This block is used to play audio messages. These messages can be TTS (Text to Speech), Audio Files (previously uploaded in Audio Resources page), or variables played as TTS.

Specify prompts to be played

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td></td>
<td>Transferring. Please be patient. Your est..</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>TTS</td>
<td></td>
<td>EstimatedWaitTime</td>
<td>text</td>
<td></td>
</tr>
<tr>
<td>TTS</td>
<td></td>
<td>minutes.</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

7. Configure the second **Play Message** block. An example is below:
8. Configure the **Assign Variables** block as shown below:

**Properties - Assign Variables**

This block can assign values of expressions to variables. Define a variable in the Initialize phase or block and select it in this block to assign it values or results of ECMAScript expressions. You can also call ECMAScript utility functions, such as sorting an array, and provide an input to be run through the function.

**Assignments**

String values must be surrounded by single quotes.

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 $\text{sdata(target, statistic)}$ or $\text{callage()}$. Strings must be enclosed in single quotes. For example:

```
Threshold Expression: 'callage() < ' + myvar
```

For more information about using ECMAScript in Designer, see ECMAScript Expressions.

**Important**

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

Extensions

Use this section to add an extension as a key-value pair to this block.
Use JSON format

This option allows certain special characters to be used in extensions, such as full stops (.), single quotes ('), double-quotes ("), and At signs (@).

Add Extension Data

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.

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

Contents

• 1 Call Routing tab
  • 1.1 Select Routing Type section
  • 1.2 Other Routing Settings section
  • 1.3 Example
  • 1.4 Ring on No Answer scenarios
• 2 Skill Expression tab
• 3 Treatments tab
  • 3.1 Message
  • 3.2 Shared Module
• 4 Routing Priority tab
  • 4.1 Use Priority during Routing
  • 4.2 Increment Priority every ___ seconds
  • 4.3 Limit Priority to
• 5 Advanced tab
  • 5.1 Targeting section
  • 5.2 Greetings section
  • 5.3 Extensions section
  • 5.4 Post processing application section
• 6 Results tab
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 interaction 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 interaction. 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 interaction.
  - **any skill** - The application can use any of the selected skills to route the interaction.

**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 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.
• **Use Skill Proficiency level** - Enter a Skill level. The interaction 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 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.

**Agent Group routing**

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

**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`.

**Campaign Group routing**

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

**Route to another Application**

Transfer the interaction to another application. If you select this option, you must first select the target **Application**. You can then select a **Phone Number** that is associated with the target application, or a variable that contains the name of a **Route Point**. If you select a variable, Designer obtains the **Route Point** from its value, but still requires that a target **Application** be selected. This is to accommodate chat interactions for applications that are enabled for omni-channel support, even if the application is not yet enabled for this capability.

**Direct number routing (voice only)**

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.

**Important**

- Direct number routing works for external transfers only.
- Genesys recommends using **Force route (voice only)** instead of this option.
Force route (voice only)

Force the call to route to a direct number. When selected, you can specify the target as a *literal* value, or as a variable that holds a *string*, *number*, or *object* value. When invoking an external transfer, use the following format: `phone number@.DN`

**Important**

When using **Force Route**:

- 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.
- The overall timeout for the **Route Call** block is limited to 30 seconds.

Parking Queue (digital interactions only)

If a digital interaction arrives when the business is closed, send the interaction to a parking queue until the business is open. If selected, you can then specify a variable that tells Designer how long the interaction is to be parked (for example, the variable that holds the number of minutes until the business is open). When regular business hours resume, the interaction is retrieved and processed.

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.

**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
    - Uses

- **Select 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

**Other Routing Settings**

- Expected Waiting Time: 30 seconds
- Use Minimum Value:

After this time, processing will move on to the next block.
Ring on No Answer scenarios

A Ring on No Answer (RONA) scenario occurs when the call is routed to an agent but the agent does not answer. These types of scenarios are handled automatically by Designer, as follows:

- The agent who did not answer is placed into a Not Ready state.
- The call is returned to the queue, but the customer does not lose their place in the queue.
- The call is then routed to the next available agent.

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, send a text message (for digital channels), and/or execute a shared module.
Learn more about busy treatments

If your application supports digital interactions, you can use the **Wait** setting to specify the number of seconds that will elapse between each treatment iteration (this setting is ignored for voice interactions).

**Important**

- When routing chat interactions, Designer waits for 100 milliseconds before starting busy treatments for chats (i.e. in-queue messages). This significantly improves the accuracy of Estimated Wait Time (EWT) and Position in Queue (PIQ) values that are typically used by these messages when providing updates to customers, but may also result in the application not sending busy treatment chat messages if routing completes almost immediately (i.e. within 100 ms).
- 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.)

**Message**

Click **Add Message** 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. For digital channels, you can specify a text message to send to the customer, and use the **Wait** setting to specify how long to wait before the message repeats.

**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. For an example, see Example of a Shared Module treatment.

**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

Properties - Route Call

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  |  Skill Expression  |  Treatments  |  Routing Priority  |  Advanced

Results

- Use Priority during Routing
- Increment Priority every 30 seconds
- Initial Priority 50
- Priority Increment Size 5
- Limit Priority to 100

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.

<table>
<thead>
<tr>
<th>Segment</th>
<th>Initial Priority</th>
<th>Increment Size</th>
<th>Maximum Priority</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>100</td>
<td>20</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>Silver</td>
<td>80</td>
<td>15</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>Bronze</td>
<td>60</td>
<td>10</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

If you enable the other check box beside the field, you can select a variable for this option.

Important

If you are using priority routing (and especially if you are incrementing priority), keep in mind that if the priority of some calls is incremented while others are not, the lower
priority calls might get stuck in the queue. In addition, default routing typically uses a priority of 10, so if certain calls are default routed and other calls have a higher priority, the default routed calls will be serviced last.

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).

Important
Designer performs periodic checks to determine if any agents are logged in. Therefore, it can take up to 60 seconds before the application actually exits the block.

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.
• 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 when used with the following functions:

• **sdata**(<target>, <statistic>) - This function applies additional routing criteria based on a statistic. You can use it to specify a target and a predefined statistic, such as **StatPositionInQueue**, **StatCallsWaiting**, or **StatCallsCompleted**. For example, this expression would route an interaction to a particular Agent Group when there are more than two agents ready and available:

  sdata(Group2.GA, StatAgentsAvailable)>2

  You can also combine multiple **sdata** functions using a pipe (|) as an **OR** operator or an ampersand (&) as an **AND** operator. For example:

  sdata(VQ_1@.Q,StatServiceFactor)>30 & sdata(VQ_2@.Q,StatServiceFactor)>40

  For a list of common statistic types used by Designer, see Statistic Types. Note that when adding a statistic to your expressions, you must use the **Stat** prefix, such as **StatTimeInReadyState**, **StatAgentsAvailable**, **StatEstimatedWaitingTime**, and so on.

• **callage()** - This function returns the age of an interaction in seconds. You can use it when building an expression that specifies a time-based routing condition, such as routing an interaction if it is older than 60 seconds:

  callage()>60
Important

• For expressions that reference a variable, the callage function is treated as a string. Therefore, you must enclose the function and its operator in single quotes (e.g. 'callage() variable).

• For routing types that have multiple targets (such as Agent Group or Agent), the script defined in Threshold Expression applies to all targets.

For more information about using ECMAScript in Designer, see ECMAScript Expressions.

Override DN

(Designer on private edition only.) When an agent is selected by the Route Call block, instead of routing to the agent Designer can Force Route the interaction to the specified DN. Information about the agent selected, including the agent’s switch, is attached to the interaction as call data.

It is intended that the DN specified is a Routing Point with a strategy loaded that can parse the call data for the selected agent information and complete the transfer to the remote switch.

Important

This feature requires multisite routing to be enabled.

Greetings section

Important

The options in this section are for Voice channels only.

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.

Things to note:

- **Customer Greeting** and **Agent Greeting** are for voice calls only (not supported for chats).
- 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

Use this section to add an extension as a key-value pair to this block.

Use JSON format

This option allows certain special characters to be used in extensions, such as full stops (.), single quotes ('), double-quotes ("), and At signs (@).

Add Extension Data

Click **Add Extension Data** to add an extension. 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.

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:
Post processing application section

Use this section to specify a Digital type application that will apply post-processing after the agent has marked the interaction as done. For example, you could specify an application that provides a survey.

Important

Digital type post-processing applications cannot be used with voice calls. For Default type applications that are enabled for omnichannel support, this option is supported only for digital interactions (i.e. chats).

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

Contents

• 1 Agent Loading
• 2 Agent Occupancy
  • 2.1 How it works
• 3 Expected Waiting Time
• 4 Load Balance
• 5 Most Skilled Agent
  • 5.1 How it works
• 6 Randomly
• 7 Round Robin
• 8 Time in Ready State
Learn about the different types of routing algorithms that you can select in the Route Call and Route (Digital) blocks.

**Related documentation:**

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 \((1 \div (1+2))\).

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

Contents

• 1 Routing Tab
  • 1.1 Select Routing type
  • 1.2 Other Routing Settings

• 2 Treatments tab

• 3 Routing Priority tab
  • 3.1 Use Priority during Routing
  • 3.2 Increment Priority every ___ seconds
  • 3.3 Limit Priority to

• 4 Advanced tab
  • 4.1 Targeting
  • 4.2 Extensions section
  • 4.3 Post processing application section

• 5 Results tab
Route Digital Block

Use this block to route digital interactions to agents based on various routing criteria.

Related documentation:

You can use the **Route Digital** block in the **Assisted Service** phase to route digital interactions to an agent based on various criteria, such as Skills and Agent Groups. This block is available for Digital application types.

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.

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.

**Route to another Application**

Route the interaction to the interaction queue of the selected destination application.

**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.

**Last Called Agent Routing**

Select this option to route the interaction to the last agent that was helping the customer. If you select this option, you must also select the **Enable Customer Profile** option in the Contextual Data tab of the Application Settings.

**Parking Queue**

If a digital interaction arrives when the business is closed, send the interaction to a parking queue until the business is open. If selected, you can then specify a variable that tells Designer how long the interaction is to be parked (for example, the variable that holds the number of minutes until the business is open). When regular business hours resume, the interaction is retrieved and processed.

**Universal Queue (IWD)**

Select this option to send interactions to the Universal Queue in Intelligent Workload Distribution (IWD) Workload Manager.

**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:

![Image of Treatments tab](image)

**Treatments tab**

Use this tab to specify a busy treatment to execute while waiting for an agent to become available. For example, the busy treatment can be a **Chat** message that you can set to repeat at regular intervals.
From this tab you can:

- Select **Repeat treatments** to repeat the treatment at regular intervals. You can then specify the interval time (in seconds) or select a variable that holds this value.
- Click **Add Chat** to add a chat treatment.

When you select a treatment, the corresponding block is automatically added to the application flow below the **Route Digital** block. For example, if you add a chat treatment, a Chat Message block is added:

You can then configure the block properties to set the desired treatment:
When routing chat interactions, Designer waits for 100 milliseconds before starting busy treatments for chats (i.e. in-queue messages). This significantly improves the accuracy of Estimated Wait Time (EWT) and Position in Queue (PIQ) values that are typically used by these messages when providing updates to customers, but may also result in the application not sending busy treatment chat messages if routing completes almost immediately (i.e. within 100 ms).

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** dropdown 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.

- Use Priority during Routing
- Increment Priority every
- Initial Priority

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

<table>
<thead>
<tr>
<th>Segment</th>
<th>Initial Priority</th>
<th>Increment Size</th>
<th>Maximum Priority</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>100</td>
<td>20</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>Silver</td>
<td>80</td>
<td>15</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>Bronze</td>
<td>60</td>
<td>10</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

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 Digital 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 when used with the following functions:

- **sdata(target, statistic)** - This function applies additional routing criteria based on a statistic. You can use it to specify a target and a predefined statistic, such as StatPositionInQueue, StatCallsWaiting, or StatCallsCompleted. For example, this expression would route an interaction to a particular Agent Group when there are more than two agents ready and available:
  
  \[
  \text{sdata(\text{Group2.GA}, \text{StatAgentsAvailable})} > 2
  \]

  You can also combine multiple sdata functions using a pipe (\(\mid\)) as an OR operator or an ampersand (\&\) as an AND operator. For example:

  \[
  \text{sdata(VQ\_1@.Q, \text{StatServiceFactor})} > 30 \quad \text{\&} \quad \text{sdata(VQ\_2@.Q, \text{StatServiceFactor})} > 40
  \]

  For a list of common statistic types used by Designer, see Statistic Types. Note that when adding a statistic to your expressions, you must use the Stat prefix, such as StatTimeInReadyState, StatAgentsAvailable, StatEstimatedWaitingTime, and so on.

- **callage()** - This function returns the age of an interaction in seconds. You can use it when building an expression that specifies a time-based routing condition, such as routing an interaction if it is older than 60 seconds:

  \[
  \text{callage()} > 60
  \]
Remember that strings must be enclosed in single quotes. For more information about using ECMAScript in Designer, see ECMAScript Expressions.

**Important**

- For expressions that reference a variable, the `callage` function is treated as a string. Therefore, you must enclose the function and its operator in single quotes (e.g., `callage()` variable).
- 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 interaction 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 Digital** 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 Digital** 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.

Extensions section

Use this section to add an extension as a key-value pair to this block.

**Use JSON format**

This option allows certain special characters to be used in extensions, such as full stops (.), single
quotes (‘), double-quotes ("), and At signs (@).

Add Extension Data

Click Add Extension Data to add an extension. 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.

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

If you add an extension, make sure to define the Key and Value settings. Otherwise, Designer displays an error.

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

Post processing application section

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 Send 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.

**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 Digital** 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 Digital** block execution.
Start Treatment Block

Contents

• 1 Module tab
  • 1.1 Is Synced
• 2 Signature tab
• 3 Navigation tab
• 4 Example
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):
Important
Do not update system variables in the Assisted Service phase while an asynchronous **Start Treatment** is running. Instead, update system variables before the Start Treatment starts or within the Self Service treatment itself.
Transfer Block

Contents

- 1 Transfer tab
- 2 Results tab
Transfer Block

- Administrator

This block enables you to transfer a call to another destination.

**Related documentation:**

- Important
  This block is only applicable to certain applications and deployments. If you do not see this block in the Palette, it is not applicable to your deployment.

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

Contents

• 1 Using this block
  • 1.1 Voice mail routing
Voice Mail Block

- Administrator

Use this block to route calls to voicemail.

**Related documentation:**

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.

**Properties - Voice Mail**

- **Voice mail routing**

  - **Voice mail box number (string value)**
  
  This block is used to route calls to a voice mail box number.

  [Select voicemail box]
Predictive Routing Block

Contents

• 1 How it works
• 2 Using this block
  • 2.1 Call Routing tab
  • 2.2 Skill Expression tab
  • 2.3 Routing Priority tab
  • 2.4 Advanced tab
• 3 Example
  • 3.1 First attempt
  • 3.2 Second attempt
  • 3.3 Third attempt
  • 3.4 Fourth attempt
If your site is enabled for Genesys Predictive Routing, you can use the Predictive Routing block to route an interaction to the agent with the highest likelihood of a successful outcome.

**Related documentation:**

- Administrator

If your site is enabled for Genesys Predictive Routing, you can use the Predictive Routing block to route an interaction to the agent with the highest likelihood of a successful outcome.

**Important**

You can only use this block if your environment is enabled for Genesys Predictive Routing. Contact your Genesys representative if you have any questions about using this feature.

The **Predictive Routing** block enables a Designer application to retrieve a list of agents that are best equipped to handle a specific type of call and then route the call to the agent with the highest likelihood of a successful outcome.

Typically, you would add this block when you want to improve routing metrics for an application that is using traditional (or "prescriptive") routing methods and has an established amount of historical data that can be used for generating accurate predictors and models.

**How it works**

Genesys Predictive Routing (GPR) uses machine learning to analyze your accumulated history of agent, customer, and call data to generate scores for each available agent that indicate probable outcomes for incoming calls. Using these scores, you can ensure that calls are routed to agents with the highest likelihood of producing a satisfactory result.

When you add the **Predictive Routing** block to an application, you can select a *Predictor* from the list of Predictors that Designer retrieves from GPR. The Predictor indicates the metric you want to optimize (such as first-contact resolution, sales conversion, and so on). The Predictor also specifies which customer and agent qualities (or *features*) have the most impact on the chosen metric.

When a call arrives, Designer retrieves a list of currently-available agents (ranked according to the parameters specified in the Predictor), and then routes the call to the best available match. Designer continues to periodically retrieve these lists of agents for the duration of the session—even if the application has moved on from the Predictive Routing block to the next routing block—until the call is finally routed or the session ends. If the "predicted" outcome differs from the actual outcome, Predictive Routing "learns" from this result, improving the future accuracy of its predictions.
Predictive Routing Block

For more information about setting up Predictors, see Creating and updating Predictors in the Genesys Predictive Routing Help.

**Important**

Only Predictors with active Models are available for selection. See Configuring, training, and testing models in the Genesys Predictive Routing Help for more information.

Using this block

This block can be placed in the Assisted Service phase of an application. For the block to take effect, you'll also need to enable predictive routing in the application settings.

**Tip**

Genesys recommends that you use this block in a mixed-use setting, where Predictive Routing is attempted first (say, for a period of 2 seconds) and then traditional routing acts as a fallback method of routing the call. This also helps to train the models used by Predictive Routing, which can effectively "learn" from the traditional routing outcomes.

**Call Routing tab**

In the **Targeting** section, select the **Predictor** you want Designer to use. You can also choose to specify this value using a variable.

In the **Evaluate** field, enter the number of agents to submit as a batch for routing, and how often to submit each batch. For example, you might choose to submit a batch of 50 agents every 10 seconds.

You can then specify additional criteria to be used when evaluating possible agents, such as their state or skills, and select the **virtual queue** for routing.

In the **Other Routing Settings** section, you can specify which routing algorithm to use and the block **timeout** value.
Skill Expression tab

Use this tab to specify a skill expression for the **Predictive Routing** block. The skill expression consists of a list of skills for which you must individually set an operator and an integer value.

**Important**

When you configure a skill expression in the **Skill Expression** tab of the **Predictive Routing** block, the **Consider Only** option (for agent status) is enabled on the **Call Routing** tab. Also, the built-in agent status filter is ignored if you use a variable-defined skill expression.

For more information about setting up skill expressions, see the Skill Expression settings for the **Route Call** block.

Routing Priority tab

For information about these settings, see the Routing Priority settings for the **Route Call** block.

Advanced tab

Greetings section

Enable the check box beside **Customer Greeting** and/or **Agent Greeting** to play an audio file to that person when the interaction is connected.
For more information, see the Advanced tab settings for the **Route Call** block.

**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.

For more information, see the Advanced tab settings for the **Route Call** block.

**Predictive Settings section**

If desired, you can specify key-value pairs that will provide additional context to the prediction service.

---

**Tip**

The **Predictive Routing** block does not have an option for treatments, but you must specify a treatment to play during routing. Genesys recommends that you do the following:

- Place a Start Treatment block and a Self Service module ahead of this block to provide a treatment to play during routing.

- In the settings for the **Start Treatment** block, disable the **Is Synced** option so that Designer will start the treatment and then immediately move on to the next block while the treatment keeps playing. This ensures that the same treatment continues for any prescriptive routing blocks that follow.
Data Blocks

Contents

- 1 Call Data
- 2 Statistic
- 3 Transaction List
Data Blocks

- Administrator

The Data blocks help you to set up and manage various data handling functions within an application.

**Related documentation:**

These blocks are used for various data handling functions, such as retrieving information about an interaction that was created before the application started running, saving some information about a customer (or an interaction) 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.
Data Blocks

Used in: Initialization, Assisted Service
Call Data Block

Contents

• 1 Using this block
  • 1.1 Use fixed key names
  • 1.2 Restricted variable names
  • 1.3 Secure and sensitive data
  • 1.4 Syntax requirements
• 2 Read Data tab
• 3 Edit Data tab
  • 3.1 Use this data for async chat (does not apply to voice)
  • 3.2 Self Service phase
  • 3.3 Assisted Service phase
• 4 Advanced tab
• 5 Examples
  • 5.1 Self Service
  • 5.2 Assisted Service
  • 5.3 Using JSON objects to update or remove key values
  • 5.4 Adding or removing large numbers of key-value pairs
• 6 Restricted variable names
Call Data Block

- Administrator

Use this block to read and write data that can be accessed from both inside and outside the application.

**Related documentation:**

The **Call Data** block enables you 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). For every **key** in the call data, you can associate a **value** with it. This value can be represented as an integer, character string, binary (boolean) type, or as a list of KVPs.

While data in application variables only exist within the application, information stored in call data can persist and be retrieved even after the application has stopped running. This allows an application to retrieve information about an interaction that was created before the application started running or preserve information about a customer or interaction that can be processed later by another application.

**Using this block**

You can add the block to the **Self Service** or **Assisted Service** phase, but the block can behave differently depending on the phase in which it is used. These differences are noted where applicable, so be sure to review the information carefully when setting up the block.

In addition, the following recommendations and guidelines must be followed when using this block:

**Use fixed key names**

Make sure to use fixed key names that won't change based on time, date, a varying ID of any kind, ANI, DNIS, or any other variables that can change between successive calls or digital interactions handled by the application. For example, avoid a key name like `myKeyName + ANI`, where ANI is the system variable that is automatically set by Designer.

**Warning**

Using dynamically changing key names is **not recommended**. They can cause significant performance issues and may be blocked in a future version of Designer.
Restricted variable names

There are certain variable names used by Designer that must not be updated by the Call Data block. See Restricted variable names for a list of these names.

Secure and sensitive data

**DO NOT** attach sensitive data, such as personally identifiable information (PII) or secure variables, to user data in the Call Data block. Otherwise, this information is captured by platform logs and reported in Designer Analytics.

Syntax requirements

When entering values, you must escape quote characters by using a preceding backslash. For example:

- **Correct:** Joe\'s Pizza
- **Incorrect:** 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

Use the **Edit Data** tab to specify key-value pairs to be written to the call data. There are some differences in how this block works, depending on whether it is added to the Self Service or Assisted Service phase. Make sure to review the details for each phase, below, so that you are aware of these differences.

Use this data for async chat (does not apply to voice)

If enabled, this option allows the key-value pairs to be carried forward from the current chat session and used later in asynchronous chat sessions that start while the interaction is still being processed. For example, if a customer starts a chat session and is transferred to an agent, Designer can retrieve the KVPs from
that session if the customer resumes activity in the chat window after the initial session timed out. This option is disabled by default.

Self Service phase

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 (see an example). When used in **Self Service**, the **Call Data** block has the following restrictions:

- Non-variable key names must follow standard JavaScript rules for variable names, even if the keys are not variables. For example, key names must only contain alphanumeric characters (no spaces) and not match any of the restricted variable names.
- Call data cannot be deleted. In the **Self Service** phase, the **Edit Data** tab does not have the **Remove** option, as is available in the **Assisted Service** phase.

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.

You can specify *literal* or *variable* values (or a combination) for each KVP, using one of the following options:

**Variable Key + Variable Value**

If you enable the variable checkbox for **Key**, the variable checkbox for **Value** is automatically selected and its dropdown menu is disabled (i.e. grayed out). Whichever variable you select for **Key** is also applied to **Value**. In this case, Designer will update the **Key** in the call data with the name of the variable and the **Value** with the value that is stored in it:

<table>
<thead>
<tr>
<th>Variable?</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>ANI</td>
</tr>
<tr>
<td>Value</td>
<td>ANI</td>
</tr>
</tbody>
</table>

**Literal Key + Literal Value**

You can also enter literal values for both **Key** and **Value**. In this case, the entries are not required to match; each value is written to the call data as specified:

<table>
<thead>
<tr>
<th>Variable?</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Department</td>
</tr>
<tr>
<td>Value</td>
<td>Sales</td>
</tr>
</tbody>
</table>
Literal Key + Variable Value

If using a literal value for Key and specifying a variable for Value, make sure that the name of the variable you select does NOT match the entry provided for Key. Otherwise, the call data won’t be updated correctly.

<table>
<thead>
<tr>
<th>Variable?</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>DialedNumber</td>
</tr>
<tr>
<td>Value</td>
<td>DNS</td>
</tr>
</tbody>
</table>

Assisted Service phase

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 or remove a list of key-value pairs in a single operation with a JavaScript (JSON) object. For more information, see Using JSON objects to update or remove key-value pairs.

If you need to update a large amount of key-value pairs (i.e. 30 or more), see Adding or Removing Large Amounts of Key-Value Pairs.

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.

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 (see Syntax requirements).
This example shows a few different ways that key-value pairs can be added as extensions:

### Examples

This section contains a few examples of how you can use the Call Data block.

- Using the block in Self Service
- Using the block in Assisted Service
- Using JSON objects to add or remove key-value pairs
- Adding or removing large numbers of key-value pairs

### Self Service

In this example, we'll use the Call Data block to specify a key-value pair to be written to the call data. We've created a variable called CustomerSegment that Designer will use to store the segment detail about the customer (for example, Bronze) and we want to associate its value with the segment key in the call data.

On the Edit Data tab, we'll click Add Data to specify a new key-value pair. For the Key field, we'll enter segment. For the Value, we'll enable the Variable checkbox and select the CustomerSegment variable we created earlier.
Watch:

For more information about using the **Edit Data** tab in the Self Service phase, see Self Service phase.

**Assisted Service**

This example 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 key-value pair (in this example, we are using `var_kvp`):
Next, initialize a variable as a list of key-value pairs using the Assign Variables block. In this example, we've done this using ECMAScript on the **Advanced Scripting** tab:

```javascript
var_kvp = {'firstName': 'Jhon', 'lastName': 'Smith'};
```

Finally, add the user data as a key-value pair in the Call Data block:
Using JSON objects to update or remove key values

You can use the Advanced Scripting tab of the Assign Variables block to create a JSON object that can update or remove key values in a single operation. In this example, we've created a JSON object that will update the value for one key to Sales and delete the value for another by assigning it a value of undefined:
You can then use the **Call Data** block to reference this JSON object and perform the operation. On the **Edit Data** tab, select the variable that contains the JSON object:

This method can also be used to make bulk updates to a large amount of key-value pairs (i.e. 30 or more), as described in the next example.

**Adding or removing large numbers of key-value pairs**

To add or delete several (i.e. 30 or more) key-value pairs, use the Advanced Scripting tab of an **Assign Variables** block to create a JSON object that lists each of the keys you want to modify.

For example, to remove a large amount of key-value pairs, create a script that assigns each key a 'null' (or undefined) value. In this example, we've defined an object called `jsonData` and assigned each of its keys a value of `null`:

```
jsonData = {
    "key1": null,
    "key2": null,
    "key3": null,
    "key4": null
}
```

If you wanted to add/update the keys, simply use the script to assign the desired values to each key. You can then reference this JSON object as a variable in the **Call Data** block to perform the desired operation:
Note that the operations are processed in smaller batches, to prevent too many requests from being processed at one time.

**Tip**
This method can be used in conjunction with the standard key-value pair operations described in Edit Data tab

**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
Warning

The addition, updating, and removal of Call Data do not happen instantaneously while the application is running. 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.

Thus, there is no guarantee that the "write" operation has finished—a subsequent "read" operation could potentially give you either an old value or an updated value on different runs through the application.

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. 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.
Statistic Block

Contents

• 1 Using this Block
• 2 Statistic Types
  • 2.1 Agent Loading
  • 2.2 Agent Loading Media
  • 2.3 Agent Occupancy
  • 2.4 Agent State
  • 2.5 Agents Available
  • 2.6 Agents Busy
  • 2.7 Agents in Queue Login
  • 2.8 Agents in Queue Ready
  • 2.9 Agents Total
  • 2.10 Calls Answered
  • 2.11 Calls Completed
  • 2.12 Calls in Queue
  • 2.13 Calls Waiting
  • 2.14 Estimated Waiting Time
  • 2.15 Expected Waiting Time
  • 2.16 Load Balance
  • 2.17 Position in Queue
  • 2.18 Service Factor
  • 2.19 Time in Ready State
• Administrator

Use this block to retrieve statistics that can be used for routing interactions.

**Related documentation:**

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, @.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 Rule 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:

$$LB =\begin{cases} 10,000,000,000 & \text{If } (ALI=0) /\text{no agents at all, wait forever}/ \\ 10,000,000,000 & \text{Else if } (AR}$$
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), 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 \( \frac{1}{4} \times (N-M) \), 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 \( \frac{3}{4} \times Y \) 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 \( \frac{1}{8} \times (N-M) \).
  - 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 \( \frac{1}{8} \times (N-M) \) 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 \( \frac{1}{4} \times (N-M) \) 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

Contents

- 1 Using this block
- 2 Example
Transaction List Block

Use this block when you want to assign transaction list values to variables.

**Related documentation:**

Using this block

You can use the **Transaction List** block when you want your Designer application to lookup a certain value in a transaction list and store it in a variable.

To use this block, add it to your application and click **Add Assignment**. Select the variable that will hold the assigned value, and then specify the details for the transaction list value you want Designer to lookup.

**Tip**

Using a **Transaction List** block can be useful in certain scenarios. But for more advanced database handling in Designer, Data Tables are recommended.

Example

In this example, we want our application to obtain a password value from a transaction list.

We've defined a variable called **TxListValue** to hold this value. After adding the
block to the application, we select that variable and provide the details for the transaction list value we want to assign it. In this case, we've told Designer to assign it the value for password in the APIKey section of the CaseLookupAPI transaction list.

Tip
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}).

Here is how our example lookup request maps back to the actual transaction list resource (you may not have access to view or modify transaction lists, so this is just provided for informational purposes):

Notice that the password value is encrypted (i.e. masked by asterisks). If the Key of a transaction list option is specified as being a password, the Value of the field is automatically hidden. However, Designer is able to read these encrypted values and use them in applications, while maintaining the secure handling.

To learn more about how Designer handles secured data in variables, see Securing Variables.
External Services Blocks

Contents

• 1 Custom Service
• 2 HTTP REST
The External Services blocks enable 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.

**Related documentation:**

Use the links below to learn more about each External Services 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

Contents

- 1 Service Details tab
  - 1.1 Input Parameters
  - 1.2 Output Parameters
- 2 Results tab
  - 2.1 Examples
Use this block to access a custom service that was created for you by Genesys.

**Related documentation:**

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**
Custom Service Block

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.

```json
{
    "thing": {
        "otherthing": "abc"
    },
    "arrayofthings": [
        "thing1", "thing2"
    ]
}
```

<table>
<thead>
<tr>
<th>JSON Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing.otherthing</td>
<td>abc</td>
</tr>
<tr>
<td>arrayofthings[1]</td>
<td>thing2</td>
</tr>
</tbody>
</table>
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

Contents

- 1 Service Details tab
  - 1.1 Input Parameters
  - 1.2 Output Parameters
- 2 Authentication tab
- 3 Results tab
- 4 Advanced tab
- 5 Test tab
- 6 Scenarios
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. The block also supports HTTPS for web services that are using server-side authentication.

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 converts a monetary value from one currency to another, or 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.

If you want to post the results of a recording captured by the Record Utterance block to the specified URL, you can specify the variable that holds a recorded file in the **Upload Record Utterance** field. (This option is only supported in the Self Service phase, as the recording file captured by the Record Utterance block is no longer available after the Self Service phase.) If you enable this option, you can use the **Recording Handle Name** option to select a file handle name to use for the audio file to be added to the upload request. If you don't make a selection, the default file handle name (**recordfile**) is used.

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:

```json
{
    "thing": {
        "otherthing": "abc"
    },
    "arrayofthings": [
        "thing1", "thing2"
    ]
}
```

<table>
<thead>
<tr>
<th>JSON Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing.otherthing</td>
<td>abc</td>
</tr>
<tr>
<td>arrayofthings[1]</td>
<td>thing2</td>
</tr>
</tbody>
</table>
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:

![Diagram of application flow with Error Handler block highlighting](image)

**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

Contents

- 1 Activity
- 2 Debug
- 3 Milestone
The Reporting blocks help you to manage certain reporting functions within an application, such as to start or stop an activity or indicate the progress of an application.

**Related documentation:**

- **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

Contents

• 1 Start tab
• 2 Stop Tab
Add this block to the Self Service or Assisted Service phase of an application when you want to start or stop activity in a report.

**Related documentation:**

Do not use **Activity** blocks for modules, as Designer reports module activity automatically. If you like, you can also nest activities to provide additional details.

---

**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

Contents

- 1 Using this Block
  - 1.1 Survey
Milestone Block

- Administrator

This block enables you to mark key moments while the application is running.

**Related documentation:**

- 

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

Contents

- 1 Debug tab
  - 1.1 Capture Checkpoint
- 2 Data tab
- 3 Advanced tab
Debug Block

Use this block to debug runtime logic that isn't running as intended.

Related documentation:

You can use this block during the application development phase to define a specific checkpoint in a module or application. This is useful when you want to debug runtime logic that isn't 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 in the DEV, QA, and UAT application streams. It is ignored in the LIVE stream.

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.

Tip

When specifying the Checkpoint name, Genesys suggests using the following naming convention:
For example:

Main_HelpDesk:Read_EWT

This can make it easier to locate and view the Debug block outputs in Designer Analytics.

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 Blocks

Contents

• 1 Book ASAP Callback V2
• 2 Book Scheduled Callback V2
• 3 Callback Availability V2
• 4 Callback V2
• 5 Cancel Callback V2
• 6 Check for Existing Callback V2
• 7 Validate Phone Number
• 8 Callback VQ Watermark
Learn about the Callback blocks available in Designer.

**Related documentation:**

- Administrator

---

**Important**

Designer supports Callback for **voice calls only**. Digital interactions are not supported.

These blocks are located in the **Callback** section of the **Palette** and manage options, rules, and features for Callback. To learn more about provisioning Callback for use with Designer, see Provisioning Callback in Designer in the **Callback Administrator's Guide**.

**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.

---

**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 Blocks

Callback Availability V2
Retrieves the scheduled callback availability from Genesys Mobile Services (GMS).

Callback V2
Offers callback and reconnects to the customer when an agent is ready.

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

Contents

• 1 Before you start
• 2 Sample callback scenarios
  • 2.1 Immediate callbacks
  • 2.2 Scheduled callbacks
• 3 Call Routing tab
  • 3.1 Advanced Options - Overrides
• 4 Offer Callback tab
• 5 Connect Customer tab
• 6 Routing Priority tab
  • 6.1 Example
• 7 Advanced tab
  • 7.1 Greetings
  • 7.2 Music on Hold
  • 7.3 Reporting
  • 7.4 Survey
  • 7.5 Business Hours
• 8 Result tab
• 9 Callback Settings Data Table
  • 9.1 Parameters
  • 9.2 Web Callbacks (API)
You can use the Callback V2 block in the Assisted Service phase of a Default type application for inbound calls. This allows the caller to request a callback when the next agent is available or to schedule a callback for a more convenient time.

Designer supports the following types of callbacks (voice calls only):

- 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 also use the Callback V2 block in the Initialize phase of a Callback type application for scheduled callbacks. When used this way, this block processes the scheduled callback at the desired time.

**Important**

- Callback is supported for voice calls only. Digital interactions are not supported.
- If redirecting a caller to an application that contains IVR callback, only 1-step transfers are supported.

**Before you start**

Before using the Callback V2 block, you must first create a variable for the callback virtual queue. Then, you can use the Business Controls settings (such as Special Days, Business Hours, and Data Tables) 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.
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 its 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.
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 validation issues. 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 validation issues. 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 **Use Priority during Routing** is enabled, you can also choose to enable the **Set Agent Reservation Priority to current priority** option. This will apply the current priority of the call at the time an agent was found for the callback to the agent reservation request. If you choose not to enable this option, the default priority value of 10,000 is used.
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.

Business Hours

Enable this option to use the timezone specified in the Business Hours for the callback virtual queue. Otherwise, the timezone of the application is used.

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:
<table>
<thead>
<tr>
<th>Setting</th>
<th>Key</th>
<th>Description</th>
<th>Default Value</th>
<th>Web Callbacks (see Web Callbacks (API))</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQ</td>
<td>(section name)</td>
<td>Name of the Virtual Queue.</td>
<td>(none)</td>
<td></td>
</tr>
<tr>
<td>Immediate Enabled</td>
<td>_immediate_enabled</td>
<td>Enables (or disables) the option to offer Immediate Callbacks.</td>
<td>true</td>
<td>✓</td>
</tr>
<tr>
<td>Scheduled Enabled</td>
<td>_scheduled_enabled</td>
<td>Enables (or disables) the option to offer Scheduled Callbacks.</td>
<td>true</td>
<td>✓</td>
</tr>
<tr>
<td>Hold Enabled</td>
<td>_hold_enabled</td>
<td>Enables (or disables) the option to Hold (or Reject) a callback.</td>
<td>true</td>
<td>✗</td>
</tr>
<tr>
<td>Logged In Check</td>
<td>_logged_in_check</td>
<td>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.</td>
<td>false</td>
<td>✗</td>
</tr>
<tr>
<td>Immediate Blackout (minutes)</td>
<td>_immediate_blackout</td>
<td>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 <strong>Immediate Blackout</strong> 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.</td>
<td>60</td>
<td>✓</td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Callback Purge Time (minutes)</td>
<td>_callback_purge_time</td>
<td>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. It is important that this value is set to greater than 0. This will prevent the system from finding a target for callback before releasing the inbound call, causing disruption in the flow and default routing of the inbound call.</td>
<td>120</td>
<td>✓</td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Call Display Name</td>
<td>_call_display_name</td>
<td>Name to display for Caller-ID.</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Call Display Number</td>
<td>_call_display_number</td>
<td>Number to display for Caller-ID.</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Enable CPD</td>
<td>_cpd_enable</td>
<td>Enables (or disables) Call Progress Detection.</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>CPD Timeout (seconds)</td>
<td>_cpd_timeguard</td>
<td>Specifies the maximum time (in seconds) allowed for Call Progress Detection after the call is connected.</td>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>Dial Prefix</td>
<td>_prefix_dial_out</td>
<td>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.)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td><strong>Important</strong></td>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dial Retry Timeout (seconds)</td>
<td>_dial_retry_timeout</td>
<td>Time to wait (in seconds) before making another attempt to dial an outbound call, if the previous attempt failed.</td>
<td>30</td>
<td>✓</td>
</tr>
<tr>
<td>Max Dial Attempts</td>
<td>_max_dial_attempts</td>
<td>Maximum number of times to try dialing an outbound call.</td>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>Min Time Before Callback (seconds)</td>
<td>_min_time_before_callback</td>
<td>Minimum time (in seconds) between the disconnection of the inbound call and the dialing of the outbound call.</td>
<td>60</td>
<td>✓</td>
</tr>
<tr>
<td>Snooze Duration (minutes)</td>
<td>_snooze_duration</td>
<td>Time to wait (in minutes) before dialing a caller who chose the “snooze” option from the menu.</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>Pushed Callback Expiry Time (minutes)</td>
<td>_pushed_callback_expiry_time</td>
<td>Duration (in minutes) to keep user-originated callback sessions alive.</td>
<td>120</td>
<td>✓</td>
</tr>
<tr>
<td>Push Notification Threshold</td>
<td>_push_notification_threshold</td>
<td>Estimated duration (in minutes) for callback sessions.</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>(minutes)</td>
<td></td>
<td>minutes) before callback time for courtesy push notification to be sent.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skill Expression</td>
<td>_target</td>
<td>The skill expression to use for targeting an agent. (Example: Billing&gt;0&amp;Collections&gt;0)</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Attach Userdata</td>
<td>_attach_udata</td>
<td>Specifies the format in which the user data should be attached to the interaction before it is routed to an agent.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Selecting <strong>single_json</strong> will attach all user data as one JSON object (key: GMS_UserData).</td>
<td>separate_keys</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Selecting <strong>separate_keys</strong> will attach each user data as a separate key. (The name of the key will be the same as the user data key.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Hours</td>
<td>_business_hours_service</td>
<td>Name of the Business Hours entry for this VQ. This name must correspond to one of the entries in Business Hours.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <strong>Immediate Offer Hours</strong> is not</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Immediate Offer Hours</td>
<td>_immediate_offer_hours</td>
<td>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. However, this option uses the timezone of the Designer application.</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
### Setting Key Description Default Value Web Callbacks (see Web Callbacks (API))

- **Immediate callback will not be offered if the current time plus the Estimated Wait Time (EWT) is outside of the hours defined in Business Hours (_business_hours_service).** For example, if Business Hours are set to Monday-Friday, 09:00 - 17:00, the EWT is 10 minutes, and the current time is Monday at 16:55, then Immediate callback will not be offered.

**Important**

- **This option disables both Immediate Enabled and Immediate Blackout (minutes).**
- **If Immediate Offer Hours is configured, the override for offering Immediate callback configured in the Callback V2 block.**
<table>
<thead>
<tr>
<th>Setting</th>
<th>Key</th>
<th>Description</th>
<th>Default Value</th>
<th>Web Callbacks (see Web Callbacks (API))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Direction</td>
<td><code>_call_direction</code></td>
<td>Determines who will initiate the call to the target.</td>
<td>USERTERMINATED</td>
<td>(n/a)</td>
</tr>
<tr>
<td>Slot Capacity</td>
<td><code>_max_request_by_time_bucket</code></td>
<td>How many callbacks can be offered for each slot.</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>Slot Duration (minutes)</td>
<td><code>_request_time_bucket</code></td>
<td>Duration (in minutes) of the time slots for scheduled callbacks.</td>
<td>15</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Important</strong> 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing Point</td>
<td><code>_route_point</code></td>
<td>Routing Point (RP) to use for making the outbound call.</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Callback Application</td>
<td><code>_service</code></td>
<td>The name of the Designer callback application.</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Application Stream</td>
<td><code>_stream</code></td>
<td>Specifies the stream of the application (of Callback type) that will be used.</td>
<td>Live</td>
<td>✓</td>
</tr>
<tr>
<td>Callback TTL (seconds)</td>
<td><code>_ttl</code></td>
<td>Specifies how long (in seconds) the callback record is stored on the system.</td>
<td>259200</td>
<td>✓</td>
</tr>
<tr>
<td>Setting</td>
<td>Key</td>
<td>Description</td>
<td>Default Value</td>
<td>Web Callbacks (see Web Callbacks (API))</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>from the Desired Callback Time) and cannot be changed.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Web Callbacks (API)

If you are booking callbacks directly with the API, not all settings in the CALLBACK_SETTINGS data table are taken into account. The parameters that are included with API callbacks are marked with a checkmark (✓) in the Web Callback column. For the following settings, the API reads them, but only when the associated parameter is set to true:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Associated callbacks create API parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate Enabled</td>
<td>checkImmediateEnabled</td>
</tr>
<tr>
<td>Scheduled Enabled</td>
<td>checkScheduledEnabled</td>
</tr>
<tr>
<td>Immediate Blackout</td>
<td>checkWithinImmediateOfferTime</td>
</tr>
<tr>
<td>Business Hours</td>
<td>checkWithinImmediateOfferTime</td>
</tr>
<tr>
<td>Immediate Offer Hours</td>
<td>checkWithinImmediateOfferTime</td>
</tr>
</tbody>
</table>

Parameters with a crossmark (✗) can be checked manually with the appropriate API calls prior to booking the callback. For more information, see Booking a callback using the callbacks create API.
Book ASAP Callback V2

Contents

• 1 Inputs tab
  • 1.1 Example
• 2 Results tab
  • 2.1 Example
Use this block to set up immediate callbacks.

**Related documentation:**

- [Callback Administrator's Guide](#)

---

**Important**

- This block relies on callback functionality provided by Genesys Mobile Engagement (formerly known as Genesys Mobile Services). Read the Callback Administrator’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

Properties - Book ASAP Callback V2

This block is used to book an ASAP Callback on Genesys Mobile Services (GMS) for a particular Callback service.

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

Properties - Book ASAP Callback V2

This block is used to book an ASAP Callback on Genesys Mobile Services (GMS) for a particular Callback service.

**Select the variables to store the results.**

The possible values for the outcome variable are:
- *BOOKED ASAP* (if booking is successful)
- *undefined* (if booking failed)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>Outcome of the query.</td>
<td>varBookCallback</td>
</tr>
<tr>
<td>Callback ID</td>
<td>ID of new callback</td>
<td>GmsCallbackServiceID</td>
</tr>
</tbody>
</table>

Results tab

Select the variables that will store the results of the **Outcome** and **Callback ID** queries.
Book Scheduled Callback V2

Contents

• 1 Inputs tab
  • 1.1 Example
• 2 Results tab
  • 2.1 Example
This block can be used to book a scheduled callback.

**Related documentation:**

- Administrator

**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

Properties - Book Scheduled Callback V2

This block is used to book an Scheduled Callback on Genesys Mobile Services (GMS) for a particular Callback service.

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

Properties - Book Scheduled Callback V2

This block is used to book an Scheduled Callback on Genesys Mobile Services (GMS) for a particular Callback service.

Inputs

Select the variables to store the results.
The possible values for the outcome variable are:
- **BOOKED_SCHEDULED**
- **undefined**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>Outcome of the query.</td>
<td>varBookCallback</td>
</tr>
<tr>
<td>Callback ID</td>
<td>ID of new callback</td>
<td>GmsCallbackServiceID</td>
</tr>
</tbody>
</table>
Callback Availability V2

Contents

• 1 Inputs tab
  • 1.1 Example
• 2 Results tab
  • 2.1 Example
This block checks if there are any available slots for the customer's preferred callback date and time.

**Related documentation:**

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**

**Properties - Callback Availability V2**

This block is used to retrieve the scheduled callback availability from Genesys Mobile Services (OMS) for a particular Callback service.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Queue</td>
<td>Callback Virtual Queue</td>
<td>variable</td>
<td>callbackVQ</td>
</tr>
<tr>
<td>Desired Day of Week</td>
<td>Desired day of week to schedule callback (0=Sunday, 1=Monday, ..., 6=Saturday)</td>
<td>variable</td>
<td>varDesiredDay</td>
</tr>
<tr>
<td>Desired Hour</td>
<td>Desired hour of day to schedule callback (0-23)</td>
<td>variable</td>
<td>varDesiredHour</td>
</tr>
<tr>
<td>Desired Minute</td>
<td>Desired minute to schedule callback (0-59)</td>
<td>variable</td>
<td>varDesiredMinute</td>
</tr>
</tbody>
</table>

**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

Contents

• 1 Inputs tab
  • 1.1 Example
• 2 Results tab
  • 2.1 Example
This block enables you to cancel a callback.

**Related documentation:**

**Inputs tab**

Provide the **Callback ID** and **Virtual Queue** of the callback to be cancelled.

**Example**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback ID</td>
<td>ID of the callback to cancel</td>
<td>variable</td>
<td>parentCallbackServiceID</td>
</tr>
<tr>
<td>Virtual Queue</td>
<td>Callback Virtual Queue</td>
<td>variable</td>
<td>callbackVQ</td>
</tr>
</tbody>
</table>

**Results tab**

Select a variable to store the results of the cancellation request.
Example

Properties - Cancel Callback V2

Cancels an existing callback.

### Inputs

Select the variables to store the results.
The possible values for the outcome variable are:
- "CANCEL_OK"
- "undefined"

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>Outcome of the cancellation.</td>
<td>checkExistingResult</td>
</tr>
</tbody>
</table>
Check for Existing Callback V2

Contents

- 1 Inputs tab
  - 1.1 Example
- 2 Results tab
  - 2.1 Example
This block checks if an existing callback already exists for a customer's phone number in the specified virtual queue.

**Related documentation:**

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Queue</td>
<td>Callback Virtual Queue</td>
<td>variable</td>
<td>callbackVRQ</td>
</tr>
<tr>
<td>Phone Number</td>
<td>The phone number to receive the callback</td>
<td>variable</td>
<td>callbackPhoneNumber</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>Outcome of the query.</td>
<td>checkExistingResult</td>
</tr>
<tr>
<td>Callback ID</td>
<td>ID of the existing callback</td>
<td>parentCallbackServiceID</td>
</tr>
<tr>
<td>Requested Date</td>
<td>Requested date of the callback in local time zone (if exists)</td>
<td></td>
</tr>
<tr>
<td>Requested Time</td>
<td>Requested time of the callback in local time zone (if exists)</td>
<td></td>
</tr>
<tr>
<td>Requested Day of Week</td>
<td>Requested day of the week (0=Sunday, ..., 6=Saturday) of the callback in local time zone (if exists)</td>
<td></td>
</tr>
<tr>
<td>Upcoming Hours</td>
<td>Number of hours until requested scheduled callback (if exists)</td>
<td></td>
</tr>
<tr>
<td>Upcoming Minutes</td>
<td>Number of minutes (excluding hours) until requested scheduled callback (if exists)</td>
<td></td>
</tr>
</tbody>
</table>
Validate Phone Number Block

Contents

- 1 Inputs tab
  - 1.1 Example
- 2 Results tab
- 3 Number Validation Configurations Data Table
  - 3.1 Parameters
Validate Phone Number Block

* Administrator

Use this block with Callback to validate phone numbers and provide support for international phone numbers.

**Related documentation:**

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*. 
Validate Phone Number Block

Example

Properties - Validate Phone Number

This block is used to validate a phone number on Genesys Mobile Services (GMS) using Google’s libphonenumber library (https://github.com/google/libphonenumber).

Validates and parses a phone number.

The phone number is validated and parsed using the Java implementation of Google’s libphonenumber library.

The input **Phone Number** can be in international format (e.g., ‘+1 650 466-1100’, ‘+441276457000’, or ‘+33 1 41 10 17 17’), or in a format recognizable within the home country specified (e.g., ‘(650) 466-1100’ in the United States, ‘(02) 276-457000’ in the United Kingdom, or ‘01 41 10 17 17’ in France), with or without spaces or common punctuations.

If **Phone Number** is specified in an international format, **Home Country Code** is not used in determining the phone number’s region and any applicable internation. If **Phone Number** is provided in national number format instead, the number is validated as a number within the country specified by **Home Country Code**. Examples:

- If **Phone Number** is ‘+1 650 466-1100’ and **Home Country Code** is ‘FR’, the number is valid.
- If **Phone Number** is ‘01 41 10 17 17’ and **Home Country Code** is ‘FR’, the number is invalid.
- If **Phone Number** is ‘01 41 10 17 17’ and **Home Country Code** is ‘US’, the number is invalid.

If the optional input **Geocoding Locale** is provided, the library will attempt to use this locale when returning the **Location** of the phone number (if it can be determined). In practice, the library does not have a comprehensive location database in different locales, and will likely return the English name for the location even when locale is set.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone Number</td>
<td>The phone number to be validated</td>
<td>variable</td>
<td>phoneNumber</td>
</tr>
<tr>
<td>Home Country Code</td>
<td>2-letter ISO country code of the home country expected, e.g., 'US', 'GB'</td>
<td>variable</td>
<td>validationDefaultCountry</td>
</tr>
<tr>
<td>Geocoding Locale</td>
<td>(Optional) The preferred locale used to return the location, e.g. 'en', 'fr-FR', 'zh-CN'</td>
<td>variable</td>
<td></td>
</tr>
</tbody>
</table>

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:
## Validate Phone Number Block

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config Name</td>
<td>Name of the configuration.</td>
</tr>
<tr>
<td>Validation Enabled</td>
<td>If number validation is enabled.</td>
</tr>
<tr>
<td>Default Country</td>
<td>The default country to be used for validation. (This should also be used for the <strong>Home Country Code</strong> in the Validate Phone Number block.)</td>
</tr>
<tr>
<td>Expects International Number</td>
<td>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 (e.g. +808) or toll-free (e.g. +800) numbers that are not the default country must be added to <strong>Additional Countries Allowed</strong>.</td>
</tr>
<tr>
<td>Always Say Country on Confirm</td>
<td>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.)</td>
</tr>
<tr>
<td>Say Country When Not Allowed</td>
<td>The country name will be stated if a phone number is not from an allowed country, or use a generic message (such as &quot;your country&quot;). Tip: You can set this option to false if you do not want your voice talent to record the names of all possible countries.</td>
</tr>
<tr>
<td>Always Says International Number</td>
<td>When confirming a phone number to the caller, always state it in full international format. For example, say &quot;15551100&quot; instead of &quot;5551100&quot;.</td>
</tr>
<tr>
<td>Premium Rate Allowed</td>
<td>Whether premium rate numbers (e.g. 1-900 numbers in the U.S.) are allowed.</td>
</tr>
<tr>
<td>Toll Free Allowed</td>
<td>Whether toll-free numbers (e.g. 1-800 numbers in the U.S.) are allowed.</td>
</tr>
<tr>
<td>Shared Cost Allowed</td>
<td>Whether shared-cost numbers (e.g. +808 numbers) are allowed.</td>
</tr>
<tr>
<td>Voicemail Allowed</td>
<td>Whether voicemail numbers (if they can be determined) are allowed.</td>
</tr>
<tr>
<td>Pager Allowed</td>
<td>Whether pager numbers (if they can be determined) are allowed.</td>
</tr>
<tr>
<td>Additional Countries Allowed</td>
<td>Select the countries, in addition to the default country, which are allowed.</td>
</tr>
</tbody>
</table>
Callback VQ Watermark Block

Contents

• 1 Inputs tab
  • 1.1 Example
• 2 Results tab
  • 2.1 Example
You can use this block to check the number of active callbacks that are currently queued for a specific virtual queue (VQ).

**Related documentation:**

This block enables you to check the number of active callbacks that are currently queued for a specific virtual queue (VQ). The result is returned as a "watermark" value that represents the number of executed callbacks.

This can be useful for certain business scenarios, such as when you want to stop offering immediate callbacks if there are a certain number of active calls already in the queue.

**Inputs tab**

Specify the Callback VQ to check for callbacks that are waiting in the queue or in a state of execution (such as being dialed or routed to an agent).

**Example**

```
Properties - Callback VQ Watermark

This block is used to check the number of callbacks in queue and being processed for a particular VQ.
```

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQ</td>
<td>Callback VQ</td>
</tr>
</tbody>
</table>

**Results tab**

Specify the variables that will store the results of the query.

The following outputs are possible:
Callback VQ Watermark Block

- **outcome** (string)
  - **WATERMARK_OK** (if query OK)
  - **undefined** (failure)
- **watermark** (the number of executed callbacks)

Example

Properties - Callback VQ Watermark

This block is used to check the number of callbacks in queue and being processed for a particular VQ.

Select the variables to store the results.
The possible values for the outcome variable are:
- **WATERMARK_OK** (successful)
- undefined (failed)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>Outcome of the query.</td>
<td>outcome</td>
</tr>
<tr>
<td>Watermark</td>
<td>ID of the existing callback</td>
<td>watermark</td>
</tr>
</tbody>
</table>
Survey Blocks

Contents

- 1 Setup Survey
Survey Blocks

These blocks help you to set up and offer surveys to customers.

**Related documentation:**

These blocks are used to manage surveys offered to customers.

**Setup Survey**

Sets up a survey for the customer.

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

Surveys are typically offered during the **Self Service** phase and completed by an accepting customer after they have finished interacting with an agent in the **Assisted Service** phase. But if you have set up your survey to be **Immediate**, the customer can complete the actual survey while still in the **Self Service** phase of the application.
Setup Survey Block

Contents

• 1 Using this Block
  • 1.1 Offer the Survey
  • 1.2 Set Up Survey Segment
  • 1.3 No Survey Segment
  • 1.4 Survey Not Offered
• 2 Creating the Survey Application
  • 2.1 Example
Use this block to set up a survey that you can offer to customers.

**Related documentation:**

**Important**
Surveys are not supported on asynchronous chat or messaging channels.

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 session, in either the **Self Service** phase or before routing begins in the **Assisted Service** phase. Then, once the customer 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 customer completes the survey within the **Self Service** phase of the current application), or after the customer has finished talking to an agent (if they agreed earlier to take the survey, the customer is then sent to a number assigned to a survey application).

You can also choose to not start the survey if the customer rejects the offer or to not offer the survey at all.

This video demonstrates how to build a chat-based survey in Designer and how you can use the Surveys dashboard to view and track the results:

**Link to video**

**Using this Block**

In most applications, you will place a User Input block before the **Setup Survey** block and use prompts to ask the customer if they want to take a survey. You can then use a Segmentation block to segment the interaction based on the customer’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 interaction 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 customer.

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 customer pressed 1 to accept the survey. Next, the application uses a Play Message block to thank the customer for taking the survey.
Next, place a **Setup Survey** block before the interaction is routed to an agent and select the **Post agent: Survey will start after talking to an agent** option. Our example is based on a voice interaction, so we’ll also 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.
For a chat-based survey, leave the DN field blank. You can specify the name of the chat survey application in the Advanced tab of the Route Call Block (or Route Digital Block), under the Post processing application section.

The interaction 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 customer 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 customer was never offered a survey. This can happen for several reasons. For example, the customer 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 customer 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 customer 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 interaction 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 customer, and survey_iAnswer1 and survey_iAnswer2 holds the answer from the customer.

Properties - Initialize
This block or phase is typically used to setup variables for the application and initialize them. Assign blocks can be used to calculate expressions and assign their results to variables in this phase.

User Variables

Specify User Variables. String values must be surrounded by single quotes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Private</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>question1</td>
<td>'Was the agent able to answer your question? Press 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>survey_iAnswer1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>question2</td>
<td>'How would you rate the agent on a scale of 1 to 5?'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>survey_iAnswer2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Editable</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>survey_sOffer</td>
<td>No</td>
<td>Specifies whether a survey was offered, accepted, or rejected. This variable is set by the Setup Survey block.</td>
</tr>
<tr>
<td>survey_iRecommendScore</td>
<td>Yes</td>
<td>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).</td>
</tr>
<tr>
<td>survey_iAgentScore</td>
<td>Yes</td>
<td>Specifies a user satisfaction score for the agent (if this question is asked in the survey).</td>
</tr>
<tr>
<td>survey_iCompanyScore</td>
<td>Yes</td>
<td>Specifies a user satisfaction score for the company (if this question is asked in the survey).</td>
</tr>
<tr>
<td>survey_iCallScore</td>
<td>Yes</td>
<td>Specifies a user satisfaction score for the entire call (if this question is asked in the survey).</td>
</tr>
<tr>
<td>survey_iProductScore</td>
<td>Yes</td>
<td>Specifies a user satisfaction score for the product (if this question is asked in the survey).</td>
</tr>
<tr>
<td>survey_sQ1..10</td>
<td>Yes</td>
<td>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.)</td>
</tr>
<tr>
<td>survey_iQ1..10</td>
<td>Yes</td>
<td>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.)</td>
</tr>
</tbody>
</table>

**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.

**Properties - Q1 - Was your issue resolved?**

This block is used to ask a question and collect input from the user. It provides options for multiple attempts.

**Specify prompts to play to collect user input**

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td>✔</td>
<td>question1</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

Prompts must finish completely before users can provide input

Timeout - wait for **5** s before assuming that no input was received.
In each **Milestone** block, select the question and answer to send to reporting.

**Properties - Q1 - Report**

This block is used to record a milestone in reports including surveys.

**Milestone**

- question1

**Milestone Type**

- Survey

**Survey Milestone Properties**

**Survey Question**

- question1

**Corresponding Answer**

- survey_iAnswer1

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

Specify prompts to play to collect user input

+ Add Prompt

<table>
<thead>
<tr>
<th>Type</th>
<th>Var?</th>
<th>Value</th>
<th>Play as</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td>☑</td>
<td>question2</td>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

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

Store output result (either DTMF entered digits, or the ASR utterance) in this variable

survey_iAgentScore
Setup Survey Block

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
Designer Analytics

Contents

• 1 Session Detail Records
• 2 Dashboards
  • 2.1 System dashboards
Learn how Designer Analytics can provide you with a deeper insight into your contact center operations.

**Related documentation:**

Designer Analytics is a powerful tool that provides a rich overview of your contact center operations. It features a series of informative dashboards, each of which offers a variety of visualizations and in-depth reporting panels that highlight specific aspects of your operations.

You can 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 can quickly answer questions you might have about your operations, such as:

- How long are customers waiting for an agent?
- How many customer interactions did we receive yesterday? Last week? Last month?
- How many of our customers are contacting us from North America? Europe? Asia?

**Key features**

- Almost real-time reporting means that as soon as an application session ends, Designer Analytics starts using the data to build reports.

*Important*

After an application session ends, it can take up to 15 minutes for the reporting data to be available on the dashboards.

- 90-day data retention, so you can see how your applications are performing over time.
- Advanced filtering options, so you can focus on the data you want to see.

**Session Detail Records**

The data contained in the Session Detail Records (SDR) is the "secret sauce" that Designer uses to generate the reporting data 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.

**Dashboards**

Designer includes several 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.

For more information about working with the dashboards, see Dashboard management.

**System dashboards**

The following system dashboards are included with Designer and ready to use. Use the dashboard icons to quickly navigate between the different types:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🏡</td>
<td>Summary provides a high-level overview of how application sessions are being handled across the platform.</td>
</tr>
<tr>
<td>☑️</td>
<td>Application Details provides how customers are moving through the application flows, such as milestones reached, activities completed, and paths taken.</td>
</tr>
<tr>
<td>⌛️</td>
<td>Durations shows how much time customers are spending in various parts of the applications.</td>
</tr>
<tr>
<td>📊</td>
<td>Data Tables displays disposition information for your applications in a spreadsheet-like table format.</td>
</tr>
<tr>
<td>Icon</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td><img src="image" alt="Spikes Icon" /></td>
<td>Spikes displays the peaks in your application session counts over a specific period of time.</td>
</tr>
<tr>
<td><img src="image" alt="Heatmap Icon" /></td>
<td>Heatmap provides the intensity in occurrence (or &quot;heat&quot;) of a particular item or event.</td>
</tr>
<tr>
<td><img src="image" alt="Path Icon" /></td>
<td>Path provides a visualization of how customers are moving through an application.</td>
</tr>
<tr>
<td><img src="image" alt="Sankey Path Analysis Icon" /></td>
<td>Sankey Path Analysis is similar to Path, but generates the results based on milestones and menu inputs.</td>
</tr>
<tr>
<td><img src="image" alt="Sunburst Path Analysis Icon" /></td>
<td>Sunburst Path Analysis renders your reporting data as a sunburst graphic.</td>
</tr>
<tr>
<td><img src="image" alt="Inputs Icon" /></td>
<td>Inputs provides information about how customers are responding to the various menu options.</td>
</tr>
<tr>
<td><img src="image" alt="Surveys Icon" /></td>
<td>Surveys provides a deeper look into the performance of your survey applications.</td>
</tr>
<tr>
<td><img src="image" alt="External Services Icon" /></td>
<td>External Services provides information about external requests made by the applications.</td>
</tr>
<tr>
<td><img src="image" alt="Routing Analysis Icon" /></td>
<td>Routing Analysis provides details about routing sessions.</td>
</tr>
<tr>
<td><img src="image" alt="Business Controls Icon" /></td>
<td>Business Controls provides details about Business Controls objects.</td>
</tr>
<tr>
<td><img src="image" alt="Bots Icon" /></td>
<td>Bots provides details about the performance of your voice and chat bots.</td>
</tr>
<tr>
<td><img src="image" alt="Session Detail Records Icon" /></td>
<td>Session Detail Records displays some of the raw data contained in the Session Detail Records (SDRs).</td>
</tr>
</tbody>
</table>
Dashboard management

Contents

- 1 Rows and Panels
- 2 Dashboard controls
  - 2.1 Time picker
  - 2.2 Refresh
  - 2.3 Home (saved default)
  - 2.4 Load
  - 2.5 Save
  - 2.6 Configure
- 3 Navigation, Query, and Filtering tabs
  - 3.1 Navigation tab
  - 3.2 Query tab
  - 3.3 Filtering tab
Dashboard management

Administrator

Learn how to manage the Designer Analytics dashboards.

Related 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.)

Some panels have a View menu that you can expand to select a visualization option or change the Interval displayed:

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. (Currently, custom queries only work on the Session Detail Records dashboard. You can use filters and drill-down options to achieve the same result on other dashboards.)

For example, you might enter the following query:
applicationName:"Test 2" AND finalDisposition:"System Error" AND ANI=5551234321
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 **5551234321**.

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:
Dashboard management

```
field: must
finalID
signature: ddfilter
query: "Abandoned in Self Service"
```

Save  Apply
Summary

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Total
  • 1.3 System Errors
  • 1.4 Abandoned
  • 1.5 Duration > 10 Mins
  • 1.6 Routing > 5/10 Mins
  • 1.7 Count by Apps
  • 1.8 Count by Disposition
  • 1.9 Count by DID
  • 1.10 Dispositions by App
Learn about the Summary dashboard and the reports it contains.

**Related documentation:**

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.

Common dispositions include:

**default** — This code is used when no other disposition code is applicable. For example, it could indicate that a call was not routed, not terminated (by any party), and was likely still active when the session and/or application completed.

**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** - — 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.
- **Auto-Stop** — the session and interaction were auto-terminated by Designer after the interaction was processed multiple times and exceeded the defined threshold, or a single application session lasted beyond the **MaxTime** defined in the application (see System Variables).

**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.

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. (See also Dispositions.)

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 gives 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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 By Disposition
  • 1.4 By User Disposition Category
  • 1.5 By User Disposition
  • 1.6 Top 10 DTMF Path
  • 1.7 Top 10 Last Milestones
  • 1.8 Top 10 Activities
  • 1.9 Top 10 Milestones
  • 1.10 Top 10 Deflection Messages
  • 1.11 Performance by Activities
The Application Details dashboard provides details about how customers are moving through your application flows.

**Related documentation:**

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 toggle the Filtering tab to see the filters that are currently being applied to the dashboard.

The drop-downs on the Filter panel only list the top 100 values for each item. If the value you are looking for isn't in the list, you can toggle the Filtering tab to expose the filter queries that are currently being applied to the dashboard. You can then edit the filter query to change the value to the one you are looking for:

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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Average Duration
  • 1.4 Average Self Service Duration
  • 1.5 Average Routing Duration
  • 1.6 Routing Phase Duration by App
  • 1.7 Self Service Duration by App
Learn about the Durations dashboard and the reports it contains.

**Related documentation:**

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 Dashboard

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Data Table
This dashboard displays the disposition data of your applications in a table.

**Related documentation:**

The **Data Tables** dashboard arranges the disposition data of your applications in a tabular format.

<table>
<thead>
<tr>
<th>UserDispositionCategory</th>
<th>UserDisposition</th>
<th>2015-12-05</th>
<th>2015-12-06</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#</td>
<td>AD(s)</td>
<td>%</td>
</tr>
<tr>
<td>Transfer</td>
<td></td>
<td>5450</td>
<td>86.2</td>
<td>61%</td>
</tr>
<tr>
<td></td>
<td>Transfer End Of Path</td>
<td>4891</td>
<td>84.6</td>
<td>55%</td>
</tr>
<tr>
<td></td>
<td>Transfer Strike Out</td>
<td>399</td>
<td>114.5</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Transfer Agent Request</td>
<td>159</td>
<td>66.2</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Transfer System Issue</td>
<td>1</td>
<td>43.8</td>
<td>0%</td>
</tr>
<tr>
<td>Abandoned</td>
<td></td>
<td>1325</td>
<td>58.3</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>Abandoned Others</td>
<td>972</td>
<td>51.5</td>
<td>11%</td>
</tr>
<tr>
<td></td>
<td>Abandoned During Hours</td>
<td>135</td>
<td>63.4</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Abandoned Out Of Hours</td>
<td>217</td>
<td>85.9</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Abandoned System Issue</td>
<td>1</td>
<td>28.2</td>
<td>0%</td>
</tr>
<tr>
<td>Self Helped</td>
<td></td>
<td>1296</td>
<td>333.2</td>
<td>15%</td>
</tr>
<tr>
<td>Deflection</td>
<td></td>
<td>846</td>
<td>77.4</td>
<td>9%</td>
</tr>
<tr>
<td>Missing</td>
<td></td>
<td>3</td>
<td>126.8</td>
<td>0%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>8920</td>
<td>117.2</td>
<td>100%</td>
</tr>
</tbody>
</table>
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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Count Over Time - By Top Regions/Countries/Languages
Learn about the Spikes dashboard and the reports it contains.

**Related documentation:**

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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Heatmap
Learn about the Heatmap dashboard and the reports it contains.

**Related documentation:**

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.)
Heatmap

(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, 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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 DNIS Disposition Path
  • 1.4 DNIS Call Type Exit Point
  • 1.5 CallType Disposition Path
  • 1.6 Call Type Disposition Exit Point
Learn about the Path dashboard and the reports it contains.

**Related documentation:**

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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Sankey Path Analysis
Administrator

Learn about the Sankey Path Analysis dashboard and the reports it contains.

**Related documentation:**

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**
Sankey Path Analysis

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

Contents

• 1 Reports on this dashboard
  • 1.1 Count Over Time
  • 1.2 Filter
  • 1.3 Sunburst Path Analysis
Learn about the Sunburst Path Analysis dashboard and the reports it contains.

**Related documentation:**

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

Contents

• 1 Reports on this dashboard
  • 1.1 No Input
  • 1.2 No Match
  • 1.3 Strikeout
  • 1.4 Report
  • 1.5 Report by Block
  • 1.6 Blocks By NoMatch/NoInput Count
Learn about the Inputs dashboard and the reports it contains.

**Related documentation:**

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**.

<table>
<thead>
<tr>
<th>Missionaries</th>
<th>Sum No Input</th>
<th>Sum NoMatch</th>
<th>Avg RC/MT</th>
<th>Avg NM/MT</th>
<th>Avg Conf</th>
<th>#OTPMT</th>
<th>#VOLUE</th>
<th>#STD/AVG</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreMA/ConsumerMenu/ConsumerMainMenu</td>
<td>7</td>
<td>9</td>
<td>0.54</td>
<td>0.60</td>
<td>0.54</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>CoreMA/ConsumerTechSupportMenu1</td>
<td>1</td>
<td>4</td>
<td>0.08</td>
<td>0.31</td>
<td>0.83</td>
<td>2</td>
<td>11</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>CoreMA/880mtsCalenSegment/CalenSegmentMenu</td>
<td>5</td>
<td>6</td>
<td>0.42</td>
<td>0.50</td>
<td>0.05</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>CoreMA/ConsumerTechSupportOfficeOfficeReportMenu</td>
<td>0</td>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
<td>0.97</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>CoreMA/ConsumerTechSupportWindowsWindowMainMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>CoreMA/CommercialMenuPage/CommercialMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CoreMA/CommercialMenuPage/CommercialTSMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.85</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CoreMA/CommercialMenuPage/CommercialTSMenu/OtherMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.07</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CoreMA/ConsumerTechSupportMenu2</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CoreMA/Proc880mgrMenu/ContrastOfFileMenu</td>
<td>1</td>
<td>0</td>
<td>0.50</td>
<td>0.00</td>
<td>0.09</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CoreMA/InfoStchModule/DynamicMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.08</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CoreMA/NEGModule/NEGMenu1</td>
<td>1</td>
<td>0</td>
<td>1.00</td>
<td>0.00</td>
<td>0.70</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CoreMA/Proc880Interiors/RayMenu</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.09</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(maingq)</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>19</td>
<td>19</td>
<td>0.25</td>
<td>0.32</td>
<td>0.00</td>
<td>6</td>
<td>47</td>
<td>5</td>
<td>59</td>
</tr>
</tbody>
</table>

**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

Contents

• 1 Reports on this dashboard
  • 1.1 Survey Count Over Time
  • 1.2 Filter
  • 1.3 Sessions Count Over Time
  • 1.4 Sessions By Score
  • 1.5 Survey Answer Distribution
  • 1.6 Sunburst Path Analysis
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

Contents

• 1 Reports on this dashboard
  • 1.1 Count of Sessions With External Requests Over Time
  • 1.2 Filter
  • 1.3 Average Request Duration
  • 1.4 Count Of Requests By Host and Service Name
  • 1.5 AVG Durations By Host, App, and Request Name
  • 1.6 External Requests Status
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 Services

External Requests Status
This report shows the status of all external requests that were made during the given time period.
Routing Analysis

Contents

- 1 Reports on this dashboard
  - 1.1 Routing Over Time
  - 1.2 Filter
  - 1.3 Routing Attempted
  - 1.4 Dispositions
  - 1.5 Successful Target Type
  - 1.6 Selected VQ
  - 1.7 Routed to LCA
  - 1.8 Timeout Block Count > 3
  - 1.9 Block Timeout Count by Outcome
  - 1.10 Skills Relaxation Steps by Outcome
  - 1.11 Average Time in Queue (in seconds)
  - 1.12 Abandoned in Queue Stats
  - 1.13 Performance Report by Applications
  - 1.14 Performance Report by Blocks within Applications
  - 1.15 Skills Path
  - 1.16 Abandoned in Queue Stats
Learn about the Routing Analysis dashboard and the reports it contains.

**Related documentation:**

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).
Routing Analysis

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.
Business Control Dashboard

Contents

• 1 Reports on this dashboard
  • 1.1 Filter
  • 1.2 Data Tables
  • 1.3 Emergency Flags
  • 1.4 Business Hours
  • 1.5 Special Days
  • 1.6 Errors
The Business Control dashboard provides near real-time metrics on the runtime usage of Emergency Flags, Business Hours Block, Special Days, and Data Tables.

Related documentation:

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.
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

Contents

• 1 Reports on this dashboard
  • 1.1 Events Over Time
  • 1.2 All Events
• 2 Searching the Session Detail Records
Learn about the Session Detail Records dashboard and the reports it contains.

**Related documentation:**

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.
Searching the Session Detail Records

You can use the **Query** tab to enter a custom search query. For example:

```
applicationName: Joules\ Coulomb\ Direct\ Sales AND finalDisposition: Abandoned\ in\ Self\ Service AND ANI: 7031231234
```

This query would search the **Joules Coulomb Direct Sales** application for sessions where the final disposition was **Abandoned in Self Service** and the ANI was **7031231234**. Note that operators (such as **AND** or **OR**) are in caps and that a backward slash (\) is needed when using terms that include spaces.

**Important**

If you don't specify the SDR field(s) you are searching, Designer only matches the search terms against the following fields: **ANI, DNIS, SessionID, InteractionID, ConnectionID, and childIxns.id**. If you want Designer to search any other field, you must specify it in the query.

You can also use operators to search for fields that do (or do not) have values. For example, to include all SDRs where the **DNIS** field contains a value, you could use the following expression in your query:

```
DNIS: *
```

In this expression, the asterisk (*) acts as a wildcard to represent any value.

Or, if you wanted to include SDRs where the **DNIS** field does not contain a value (is blank):

```
-(DNIS: *)
```

In this expression, the minus sign (-) acts as a negative operator. The asterisk (*) again acts as a wildcard to represent any value, even one that is empty.

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.
Important

💡 Available for Paging: This message might not display correctly. For example, you might see an inaccurate total provided (e.g. 0 to 2 of 2 available for paging, when it should say 1 to 2 of 2...), or asterisks (*) inserted where they shouldn’t be (e.g. 2*00*). These anomalies are caused by an issue with Kibana (the third-party engine that drives the Analytics reports) and are not related to Designer.
Bots Dashboard

Contents

• 1 Understanding the results
  • 1.1 Bot Session
  • 1.2 Utterance
  • 1.3 Intent
  • 1.4 Intent fulfillment

• 2 Reports on this dashboard
  • 2.1 Count of Bot Sessions Over Time
  • 2.2 Filter
  • 2.3 Count by Channel
  • 2.4 Count by Result
  • 2.5 Count by Bot Method
  • 2.6 Count by Provider
  • 2.7 Invoked Bot Results
  • 2.8 Average Time SS Duration
  • 2.9 Top Intents
  • 2.10 Selected Intents Details
  • 2.11 Top Bots /W No Matched Intent
  • 2.12 Number of Bot Sessions No Matched Intent
Use the Bots Dashboard to evaluate and analyze the performance of your voice and chat bots.

**Related documentation:**

You can view the **Bots** dashboard by selecting the **Bots** icon:

The reports available on this dashboard include visualizations and detailed breakdowns of your bot sessions, such as how many sessions were successful or failed, the top intents that were detected, and how many sessions had intents detected that were not successfully fulfilled.

**Understanding the results**

This section describes some of the bot-related terminology used by Designer.

In some cases, bot service providers may use similar terms but with different applications of meaning. For example, a bot services provider might consider an intent to be *fulfilled* as soon as it is detected, while Designer doesn't consider an intent to be fulfilled until the bot has collected all the information it needs to satisfy that intent.

To better understand the results displayed on this dashboard, you should be familiar with how these terms are used within Designer.

**Bot Session**

Each **bot session** represents a single conversation between a customer and the bot service that was invoked by the Bot Block.

A bot session starts as soon as the **Bot** block receives voice or chat input from the customer and ends when Designer either moves to an intent block or to an **Error Handler** block. If the same **Bot** block is executed again at a later point in the application, it is considered to be the start of a new bot session.
Bot sessions can be classified as **Success** or **Failed**:

- **Success** indicates that Designer was able to invoke the bot. A conversation with the customer took place and the bot was able to successfully identify an intent and return it to the Bot Block.
- **Failed** indicates that there was a condition that triggered the **Error Handler** block, such as Designer not being able to communicate with the bot.

**Utterance**

During a conversation, the bot will ask the customer to provide some information. Each input that the customer provides (voice or chat response) is an **utterance**. For example, it could be a request ("I want to book a hotel room") or a simple reply to a question ("Yes").

**Intent**

During its conversation with a customer, the bot attempts to identify the **intent**, or what it is that the customer wants to do. For example, the bot might detect that the customer wants to buy a ticket. It will then proceed to fill the required slots (or "entities") that are associated with that intent, such as the name of the show the customer wants to buy a ticket for, the date and time they want to attend, and so on, by asking the customer to provide those details.

**Intent fulfillment**

After the bot has successfully gathered all of the information it needs to satisfy an intent, Designer consider the intent to be fulfilled.

**Reports on this dashboard**

This dashboard contains the following reports:

**Count of Bot Sessions Over Time**

This panel displays the total number of Bot Sessions that took place over a specified period of time for both Voice and Chat channels.

You can expand the **View** menu to select a different visualization option, or use the **Interval** setting to adjust the range of time shown, such as for the last hour, 12 hours, or the previous week.
Filter

You can use the Filter panel to select specific values you want to filter the results for, such as Application or Disposition.

The drop-downs on the Filter panel only list the top 100 values for each item. If the value you are looking for isn't in the list, you can toggle the Filtering tab to expose the filter queries that are currently being applied to the dashboard. You can then edit the filter query to change the value to the one you are looking for:

Any filters you select are applied across all of the dashboards, not just the one you are viewing. You
can toggle the Filtering tab to see the filters that are currently being applied to the dashboard.

Count by Channel

This panel displays a breakdown of total counts and percentages for both Voice and Chat channel bot sessions.

Count by Result

This panel provides total counts and percentages for Bot Session results, broken down by status. For information about Success and Failed sessions, see Understanding the results.
Count by Bot Method

This panel displays a breakdown of all Bot Sessions by the method used by the bot service (default is Native).
Count by Provider

This panel displays a breakdown of all Bot Sessions by the bot services provider that was invoked (e.g. Dialogflow, Lex, Dialog Engine).
Invoked Bot Results

This panel lists the names of all bots that were invoked during the selected time period, along with the total counts and percentages for bot sessions where the attempt to invoke the bot was successful or failed. Clicking the arrow beside the bot name expands the row to show the breakdown of these totals by channel.

You can change the reporting view to show either the **Bot Sessions by Channel** or **Channel Usage by Bot Sessions**. If desired, you can export the results to a CSV file.
Average Time SS Duration

This panel displays the average duration of time (in seconds) that all application sessions spent in the Self Service phase, broken down by total counts of application sessions where bots were invoked (Bots Executed) and all other application sessions (No Bots Invoked).

You can expand the View menu to change the visualization options or select a specific Interval to display (such as 1 day). Hovering over the results displays the details for that item.

Top Intents

This panel lists the top intents for all bots that were invoked during the selected time period. An intent is detected by the bot during its conversation with the customer.

Clicking an intent expands the row to display the names of the bots that detected that intent. Other details provided in this panel include:

- **AvgDuration(MS)** — Average duration of time (in milliseconds) that the bot sessions were active, based on the period of time between the first request sent to the invoked bot service and the last response that was received.
- **SumStepcount** — Total number of requests that were sent to the bot service.
- **AbandonedInSS** — Total number of bot sessions that ended with a final disposition of Abandoned in Self Service.
- **RoutedToAgent** — Total number of bot sessions that ended with a final disposition of Routed to Agent.
- **RoutedToDN** — Total number of bot sessions that ended with a final disposition of Routed to DN.
- **AbandonedInQ** — Total number of bot sessions that ended with a final disposition of Abandoned in Queue.
If desired, you can also choose to export the results to a CSV file.

### Selected Intents Details

This panel lists details for the top selected intents, broken down by channel type (voice or chat). If desired, you can export the results to a CSV file.

#### Top Bots /W No Matched Intent

This panel provides information about sessions where a bot was invoked, but no intent was detected.

Clicking an intent expands the row to display the totals by **Channel** type. Other details provided in this panel include:

- **AvgDuration(MS)** — Average duration of time (in milliseconds) that the bot sessions were active, based on the period of time between the first request sent to the invoked bot service and the last response that was received.
- **#Intents** — Total number of selected intents.
- **#Sessions** — Total number of bot sessions.
- **#Intents/Session** — Average total of selected intents per bot session.
Details provided in this panel include:

- **AvgDuration(MS)** — Average duration of time (in milliseconds) that the bot sessions were active, based on the period of time between the first request sent to the invoked bot service and the last response that was received.

- **SumStepcount** — Total number of requests that were sent to the bot service.

- **AbandonedInSS** — Total number of bot sessions that ended with a final disposition of *Abandoned in Self Service*.

- **RoutedToAgent** — Total number of bot sessions that ended with a final disposition of *Routed to Agent*.

- **RoutedToDN** — Total number of bot sessions that ended with a final disposition of *Routed to DN*.

- **AbandonedInQ** — Total number of bot sessions that ended with a final disposition of *Abandoned in Queue*.

If desired, you can export the results to a CSV file.

**Number of Bot Sessions No Matched Intent**

This panel expands on the results provided in the above report. It displays a total of all bot sessions broken down by *Failed Bots* and bot sessions that ended with *Intent Not Fulfilled*.

For a **Failed Bot**, the bot session experienced an error that prevented the bot service from functioning properly.

For **Intent Not Fulfilled**, the bot service was not able to match the customer's input to an intent. Everything worked, but the bot was not able to determine what the customer wanted to do.
You can expand the **View** menu to change the visualization options or select a specific **Interval** to display (such as 1 day). Hovering over the results displays the details for that item.

**Important**

In some cases, the reporting results can include a value of (missing). This indicates a value was missing from one of the fields included in the data query.
Session Detail Record (SDR) Fields Reference

Contents

- 1 List of Session Detail Record fields
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. Each entry notes whether the value is set by Designer (i.e. system-generated) or provided by the application Developer.

The Category associates each field with the source of its value. For example, it indicates if the value was pulled from Session Information (captured during application runtime) or from certain types of blocks that were encountered, such as Routing or Business Controls.

You can click the arrows in the column headers to sort the items alphabetically.

**Important**

Internal Designer system variables in SDRs During an application session, Designer adds certain internal system variables at various stages. These internal variables are recorded in Designer Analytics at the end of the application, along with other system and user-defined variables. Thus, it is possible to see new variables in Analytics that are not listed in the Initialize phase block. Although these variables may appear in variable or call data objects in Session Detail Records (SDR), they are intended only for internal use by Designer and should not be used in blocks for driving application logic. These variables can change or be removed at any time, so attempting to use them in applications can affect application resiliency and cause unexpected behavior. This type of usage is not supported or recommended. If your business operations require new functionality, contact your Genesys representative.

**Tip**

To quickly check if a field is listed on this page, you can use the search function in your browser. For example, in Chrome, press CTRL + F to open the search tool.

List of Session Detail Record fields
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Category</th>
<th>Set by</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANI</td>
<td>The Automatic Number Identification is the customer's phone number (or Caller ID).</td>
<td>Call Information</td>
<td>Designer</td>
</tr>
<tr>
<td>DNIS</td>
<td>The Dialed Number Identification Service is the phone number that the customer dialed.</td>
<td>Call Information</td>
<td>Designer</td>
</tr>
<tr>
<td>applicationName</td>
<td>The name of the application, as provided by the application Developer when creating a new application.</td>
<td>Application Settings</td>
<td>Developer</td>
</tr>
<tr>
<td>applicationID</td>
<td>The unique internal ID that Designer assigned to the application.</td>
<td>Application Settings</td>
<td>Designer</td>
</tr>
<tr>
<td>ApplicationType</td>
<td>The application type, as selected by the application developer when the application was created (see creating a new application). The values are as follows:</td>
<td>Application Settings</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>• application = Default</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• eservices = Digital</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• callback = Callback</td>
<td></td>
<td></td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional custom version of the application to use for reporting purposes, as specified by the application developer on the General tab in the application settings.</td>
<td>Application Settings</td>
<td>Developer</td>
</tr>
<tr>
<td>ApplicationReportingTitle</td>
<td>Optional custom title of the application for reporting purposes, as</td>
<td>Application Settings</td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Tip**
You can quickly view an application ID by mousing over the application link on the Applications page and checking the target URL that appears in the browser window.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Category</th>
<th>Set by</th>
</tr>
</thead>
<tbody>
<tr>
<td>SessionID</td>
<td>The unique internal ID that Designer assigned to the session.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>InteractionID</td>
<td>The unique internal ID that Designer assigned to the interaction.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>InteractionCategory</td>
<td>The channel type that was used for the interaction. For example: voice, chat, or email.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>ConnectionID</td>
<td>The connection ID that Designer assigned to the interaction. For example: 016202fe11099003</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>finalDisposition</td>
<td>Disposition code indicating the status of the interaction at the time it exited the application flow. For example, this value can indicate if the interaction was routed to an agent or the customer disconnected from the call. Disposition codes can be enabled in the Finalize phase. For more information, see disposition codes.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>LanguageName</td>
<td>The language that was set for the application at the time when the interaction ended.</td>
<td>Session Information</td>
<td>Developer</td>
</tr>
<tr>
<td>@endtime</td>
<td>Timestamp to indicate</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
</tbody>
</table>
### Session Detail Record (SDR) Fields Reference

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Category</th>
<th>Set by</th>
</tr>
</thead>
<tbody>
<tr>
<td>@timestamp</td>
<td>Timestamp to indicate when the interaction started. Example:</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td>2017-03-08T01:56:12.037Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>Duration of the total session, in milliseconds (ms).</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td><strong>Tip</strong> To calculate the duration of blocks execution and exclude session wrap-up time, subtract the value of the <code>operationalOverheadDuration</code> field from this value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>operationalOverheadDuration</td>
<td>Total amount of time, in milliseconds (ms), that the application was in an idle state. Typically, this state occurs just before the session enters the Finalize phase.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>ssduration</td>
<td>Amount of time (in milliseconds) that the customer spent in the Self Service phase of the application.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>ssstart</td>
<td>Timestamp to indicate when the Self Service phase of the application started.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>ssend</td>
<td>Timestamp to indicate when the Self Service phase of the interaction ended.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>asduration</td>
<td>Length of time (in milliseconds) that the customer spent in the Assisted Service phase.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>asstart</td>
<td>Timestamp to indicate when the interaction entered the Assisted Service phase. Example: 2017-03-08T01:56:20.933Z</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>asend</td>
<td>Timestamp to indicate when the Assisted Service of the interaction ended.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Category</td>
<td>Set by</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>variables</td>
<td>List of all user and system variables that contain values.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>survey</td>
<td>List of survey-related variables. For more information, see the <code>survey_*</code> variables listed in system variables and the Setup Survey Block page.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>calldata</td>
<td>List of all call data Key-Value Pairs (KVPs) encountered by the application. For more information, see the Call Data block page.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>callAbandonedBlock</td>
<td>If the customer leaves the session (for example, hangs up), this field contains the ID of the block that was being processed when they left.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>callEndState</td>
<td>Indicates if the call ended in the Assisted Service or Self Service phase.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>callEndParty</td>
<td>Indicates the party that ended the call, i.e. Application, Caller, or System.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>callEndReason</td>
<td>Indicates the reason that the call ended. For example, it might have been routed to an agent, abandoned, triggered a Business Controls condition, such as an emergency flag, business hours, or special day exception, or the caller selected a menu option that ended the call.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>callEndType</td>
<td>Indicates the conditions under which the call was ended, i.e. Normal or Error.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>dtmfpath</td>
<td>The menu path (via the pressing of DTMF keys)</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Category</td>
<td>Set by</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>----------</td>
</tr>
<tr>
<td>menucount</td>
<td>The number of Menu blocks the customer encountered during the session.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>inputcount</td>
<td>Total number of User Input blocks the customer encountered during the session.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>blockpath</td>
<td>An ordered list of the names of all blocks that were encountered throughout the course of the interaction.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>blockidpath</td>
<td>An ordered list of IDs for all blocks that were encountered throughout the interaction. For more information about block IDs, see Using the blocks.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>blocks*</td>
<td>List of detailed information about the individual blocks that were encountered, such as:</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td>• name of the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• unique ID of the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• type of block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• duration of time that was spent inside the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• timestamps of when the application entered and exited the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inputs*</td>
<td>List of detailed information about the various User Input blocks that were encountered, such as:</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td>• name of the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• unique ID of the block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Category</td>
<td>Set by</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>externalrequests*</td>
<td>List containing detailed information about External Services Blocks that were encountered.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>milestones</td>
<td>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.</td>
<td>Session Information</td>
<td>Designer and Developer</td>
</tr>
<tr>
<td>usermilestonecount</td>
<td>Number of user-defined milestones that were hit. These are milestones that were defined in Milestone blocks or set in other blocks, such as Menu.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>userMilestones*</td>
<td>Milestones that were defined in Milestone blocks or set in other blocks, such as Menu.</td>
<td>Session Information</td>
<td>Designer and Developer</td>
</tr>
<tr>
<td>userMilestonesPath</td>
<td>Names of all milestones that were encountered by the application during the session.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>FinalUserMilestone</td>
<td>The last user-defined milestone that was encountered during the application session (if the value of usermilestonecount is greater than 0).</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>LastMilestone</td>
<td>The most recent milestone that was encountered during the session.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>activities*</td>
<td>List of activities that were encountered during the session. Activities are defined in an Activity or automatically captured</td>
<td>Session Information</td>
<td>Developer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Category</td>
<td>Set by</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>activitycount</td>
<td>when an interaction enters or exits a Shared Module. Total number of activities that were encountered during the session. Activities are defined in an Activity block or automatically captured when an interaction enters or exits a Shared Module.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>utcstart</td>
<td>Starting time of the interaction in Universal Coordinated Time (UTC). Example: 1454643744261</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>SessionType</td>
<td>The type of session. Example: inbound</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>CountryName</td>
<td>Name of the country.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>Region</td>
<td>Name of the region.</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>childIxns*</td>
<td>This field only applies to Digital type applications and contains a list of details about new child interactions that were created by the session. The details include information about the block that initiated the child interactions (i.e. blockid, blockname, and blocktype). It can also indicate if the child interaction was created by a busy treatment (isTreatment: Y).</td>
<td>Session Information</td>
<td>Designer</td>
</tr>
<tr>
<td>attributeslist</td>
<td>Object with each key-value pair (KVP) appearing as a property that is set to the KVP value. (See also: setAttributes)</td>
<td>Generic Attributes</td>
<td>Designer</td>
</tr>
<tr>
<td>businesshourserrcount</td>
<td>Number of times a Business Hours was accessed during the interaction.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>businesshoursextcount</td>
<td>Total number of external Business Hours checks that occurred within a session, if you are</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Category</td>
<td>Set by</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>businesshoursreqcount</td>
<td>Total number of requests for Business Hours checks that occurred within a session.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>emergencieserrcount</td>
<td>Total number of Emergency Flags checks within a session.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>emergenciesextcount</td>
<td>Total number of external Emergency Flags checks within a session, if you are controlling emergency modes from a web service (for example, via an HTTP Rest block.)</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>specialdayserrcount</td>
<td>Total number of Special Days checked within a session. All checks in a single Special Day block count as one check.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>specialdaysextcount</td>
<td>Total number of Special Days checked externally within a session, if you are controlling these from a web service (for example, via an HTTP Rest block). All checks in a single Special Day block count as one check.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>specialdaysreqcount</td>
<td>Total number of exceptions that were encountered when the application was performing Special Day checks. All errors encountered in a single Special Day block count as one check.</td>
<td>Business Controls</td>
<td>Designer</td>
</tr>
<tr>
<td>routingAttempts</td>
<td>Each routing attempt is captured as an entry in the array. Some of the properties captured in each routing attempt include: • blockID and</td>
<td>Routing</td>
<td>Designer</td>
</tr>
</tbody>
</table>

Session Detail Record (SDR) Fields Reference

Designer User's Guide 644
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Category</th>
<th>Set by</th>
</tr>
</thead>
<tbody>
<tr>
<td>blockName</td>
<td>• priority - value of the priority used for the last queue submit call made by this block (applies only when priority routing is enabled).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>routingBlockCount</td>
<td>Number of Routing blocks that were encountered within a session.</td>
<td>Routing</td>
<td>Designer</td>
</tr>
<tr>
<td>routingBlockTimeoutCount</td>
<td>Number of times that routing blocks timed out.</td>
<td>Routing</td>
<td>Designer</td>
</tr>
<tr>
<td></td>
<td><strong>Tip</strong> 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>routingCallHandlingType</td>
<td>Indicates the type of routing used for the call. For example: default or consult.</td>
<td>Routing</td>
<td>Designer</td>
</tr>
<tr>
<td>routingLCAAttempted</td>
<td>Indicates if Last Called Agent routing was attempted. This option can be enabled on the Agent Routing tab of the Route Agent block.</td>
<td>Routing</td>
<td>Designer</td>
</tr>
<tr>
<td>routingSkillRelaxationCount</td>
<td>Number of Routing blocks that used skill relaxation as a routing option. The required skill level is gradually reduced until a specified minimum skill level is reached, which allows you to expand the group of agents that can receive this call if other agents are busy.</td>
<td>Routing</td>
<td>Designer</td>
</tr>
</tbody>
</table>
ECMAScript Expressions

Contents

• 1 Before you start
  • 1.1 A few notes on syntax
  • 1.2 General recommendations
• 2 Scripting examples
  • 2.1 Building a Dynamic TTS Prompt
  • 2.2 Controlling the Application Flow
  • 2.3 Checking for numeric values
  • 2.4 Declaring JSON payloads
• 3 Advanced scripting in Designer
• 4 Built-in user functions
  • 4.1 setAttributes
  • 4.2 isDataTableValueValid
In certain blocks, you can use ECMAScript expressions to perform dynamic operations while an application is running. For example, you can use ECMAScript expressions to assign values to variables or perform certain functions, like sorting an array.

**Related documentation:**

**Before you start**

To use this feature, you must have a basic level of familiarity and understanding of ECMAScript syntax and rules.

Although the terms ECMAScript and JavaScript are often used interchangeably, 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.

Designer blocks that support the use of ECMAScript expressions include:

- Assign Variables Block
- Menu Option Block
- Return Block
- Shared Module Block
- Data Tables
- Activity Block
- Milestone Block

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 interpreted as a literal string.

**Important**

ECMAScript support in Designer is provided by other services in the Genesys Multicloud CX solution. Due to certain dependencies and limitations, not all
ECMAScript functions are supported when used in Designer (for example, the `toLocaleString` function is not supported at this time). For more information about ECMAScript, see the ECMAScript page in the *Orchestration Server Developer Guide*.

A few notes on syntax

Some rules and guidelines to follow when using ECMAScript in Designer:

- **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'})`.

General recommendations

- **Keep it simple.** Don't overcomplicate your code.
- Minimize the declaration of temporary variables within Advanced Scripting.
- **Use caution!** Any errors in your script can cause erratic behavior, so test your changes to make sure that your script works correctly before running it in your production environment. Designer can check your script for syntax errors, but cannot validate it nor check for runtime errors that might occur when the script is executed.

Scripting examples

Below are examples of how ECMAScript expressions can be used in Designer applications. Some tips and general recommendations are also provided.

**Building a Dynamic TTS Prompt**

You can use the Assign block to concatenate a string to be spoken by the application. For example, the expression below reads the caller’s phone number or ID:

'You are calling from ' + ANI

**Controlling 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:

\[ \text{numDays} > 7 \land (\text{isSummer} \lor \text{isAutumn}) \]

### Checking for numeric values

When using scripting to check for numeric values, always use double "equals" signs (\(==\)). This compares the numeric values no matter which data type is being used.

For example:

```javascript
var a = '1234'; // string data type
var b = 1234;  // numeric data type
var c = (a == b);
```

In this example, the value of \(c\) will be `true`, as the actual numeric values are compared, not the string and numeric data types.

However, this does not work for Boolean values. For example, the expression

```javascript
(true == 'true')
```

would not produce a result of `true`. In this case, you would need to use an expression such as the following:

```javascript
(vResult == true || vResult == 'true' || vResult)
```

### Declaring JSON payloads

Keep your code as simple as possible. For example, the following code is unnecessarily complex:

```javascript
{
    vAPIInputCRM = new Object();
    vAPIInputCRM.IT_SEARCH = new Object();
    vAPIInputCRM.IT_SEARCH.item = new Object();
    vAPIInputCRM.IT_SEARCH.item.TYPE = 'ZPH';
    vAPIInputCRM.IT_SEARCH.item.VALUE = vANI;
    vAPIInputCRM.IV_LANGUAGE = 'IT';
    vAPIInputCRM.IV_MARKET = 'IT';
}
```

This is much better:

```javascript
vHTTPInput = {};
vHTTPInput.tid = vAccountInfo.Result.participantDetails.tid;
vHTTPInput.firstName = vAccountInfo.Result.participantDetails.firstName;
vHTTPInput.lastName = vAccountInfo.Result.participantDetails.lastName;
vHTTPInput.emailAddress = vAccountInfo.Result.participantDetails.primaryEmailId;
vHTTPInput.emailType = 'ADDRESSCHANGE';
vHTTPInput.interactionDateId = vInteractionID;
```
Advanced scripting in Designer

In blocks where it is supported, you can use use the Advanced Scripting tab to compose more complex ECMAScript constructs, such as loops or multiple nested conditions.

Let's take a look at how advanced scripting could be used to assign a value to a variable. Here, some script has been added to the Advanced Scripting tab of an Assign Variables Block:

```javascript
varOrderDetails = [
  {"item": "Laptop bag", quantity: 3, backordered: false },
  {"item": "Phone charger", quantity: 2, backordered: false },
  {"item": "Super rare fish", quantity: 1, backordered: true }
];

var i; // a local variable that exists only in this script
varOrdersPrompt = ''; // use a variable defined in Initialize phase
for (i = 0; i < varOrderDetails.length; i++) {
  // 3 laptop bags . . . give a space between quantity and item name
  varOrdersPrompt += varOrderDetails[i].quantity + ' ' + varOrderDetails[i].item;
  // its odd to hear 2 of phone charger (not chargers) - lets fix that
  varOrdersPrompt += varOrderDetails[i].quantity + ' ' + 'o' + ' ';
  if (i < varOrderDetails.length - 1) {
    varOrdersPrompt += ', '; // add a comma to give TTS a short pause
  }
}
```

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 first initializes JSON data in varOrderDetails so that it becomes an array of three JSON objects. Each JSON object has properties — item, quantity, and 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.

The 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 remain available throughout this
application flow, but not in any modules this application calls (such as varOrdersPrompt).

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

**Built-in 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.

**setAttributes**

You can use this function to add arbitrary key-value pairs to the Session Detail Record (SDR):

```javascript
setAttributes(key, value)
```

- If `key` is a string, a property with `key = value` is added to the attributes object in the SDR.
- If `key` is an object, all of its properties are added as individual properties in the attributes object in the SDR. In this case, `value` is ignored.

**Example**

```javascript
// set first key value pair
setAttributes('key1', 'Success')

// add a second one using variables in the application
var varKeyName = "key3"
var varKeyValue = "value3"
setAttributes(varKeyName, varKeyValue)

// lets add an object which will overwrite "key1" and add "key2"
var myObject = { "key1" : "value1", "key2" : "value2" }
setAttributes(myObject )
```

All these statements will finally generate this data:

```javascript
attributesList {
  "key1" : "value1",
  "key2" : "value2",
  "key3" : "value3"
}
```

**Usage and limitations**

You can use this function in all phases of Default and Digital applications, in both Self Service and Assisted Service. However, you must adhere to the following rules:
• Do not log Personally Identifiable Information (PII).
• Do not log secure variables.
• Keep the data size less than 25Kb.

Warning
Reading is not supported. Use application variables when both read and write access is required.

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.
Troubleshooting

Contents

• 1 How to check your Designer version
• 2 How to capture a network log
  • 2.1 How to generate a HAR file using Chrome
  • 2.2 How to generate a HAR file using Firefox
  • 2.3 How to generate a HAR file using Safari
  • 2.4 How to generate a HAR file using Edge
• 3 Validation and publishing errors
  • 3.1 Callback Shared Modules vs. Templates
  • 3.2 Publishing failure due to parseInt() function
This page describes how to generate network logs from different types of browsers. If you are experiencing issues with Designer, this information can be used to assist with troubleshooting.

**Related documentation:**

- [How to check your Designer version](#)
- [How to capture a network log](#)

Your Designer version is displayed in the top-right corner of the interface:

![Version: 9.0.109.00.608](image)

### How to capture a network log

Most browsers allow you to capture an HTTP Archive (HAR) file that contains details about how a particular web page has been handled by the browser. If you are asked to provide a HAR file to Genesys, follow the steps for your browser:

- Chrome
- Firefox
- Safari
- Edge

**Tip**

For best results, Genesys recommends that you generate the HAR file while using your browser's **private** (or **incognito**) browsing mode.

### How to generate a HAR file using Chrome

Launch a new **incognito** window (you can do this from the Chrome settings menu, which appears as 3 vertical dots in the top right-side corner of the browser screen) and go to the Designer page where you are experiencing the issue.
Next, from the Chrome settings menu, go to More tools > Developer tools to launch the developer tools console.

In the developer tools console, go to the Network tab and select Preserve log. Once you click the box, the red circle on the left indicates that the network activity is now being captured. (If the circle is still black, you can click it to toggle capturing on.)
While the capture is running, refresh the page and try to reproduce the issue you are having with the Designer page. When you are finished, you can stop the capture by deselecting the Preserve log box, or by clicking the red circle.

Click the Export HAR button to download the file to your computer.

How to generate a HAR file using Firefox

Open a new Private window (you can do this from the Firefox settings menu, which appears as 3 vertical lines in the top right-side corner of the browser screen) and go to the Designer page where you are experiencing the issue.

Next, from the Firefox settings menu, click Web Developer and select Network to launch the developer console.
In the developer console, go to the **Network** tab and select **Persist Logs**.

Refresh the Designer page and/or perform the action that isn't working correctly. When you are finished, click the **HAR** button and select **Save All As HAR** to download the file to your computer.
How to generate a HAR file using Safari

Open a new **Private** window (you can do this from the **File** menu) and go to the Designer page where you are experiencing the issue.

Next, from the **Safari** menu, select **Preferences**. Click the **Advanced** tab and enable the **Show Develop menu in menu bar** option.
From the **Develop** menu, select **Show Web Inspector**.
Troubleshooting

In the **Web Inspector** console, go to the **Network** tab and select **Preserve Log**.

[Image of Web Inspector console with Preserve Log selected]

Refresh the Designer page and/or perform the action that isn't working correctly. When you are finished, click the **Export** button to download the file to your computer.

[Image of Network tab with Preserve Log and Export button highlighted]

How to generate a HAR file using Edge

To start, open a new **InPrivate** window (you can do this from the settings menu, which appears as 3 horizontal dots in the top right-side corner of the browser screen) and go to the Designer page where you are experiencing the issue.

Next, from the Edge settings menu, go to **More tools > Developer tools** to launch the Developer Tools console.

[Diagram of Edge settings menu with Developer tools highlighted]
In the Developer Tools console, go to the **Network** tab. You can toggle the capture on/off using the **Play** (triangle) and **Record** (square) buttons. If the record button is not red (i.e. active), click it to start capturing data.

Refresh the Designer page and/or perform the action that isn't working correctly. When you are finished, click the **Export HAR** button to download the file to your computer.
Validation and publishing errors

During a validation check, Designer typically tells you what went wrong and where to fix it. This section describes some warnings and errors that might require a few extra steps to resolve.

- Callback Shared Modules vs. Templates
- Publishing failure due to parseInt() function

Callback Shared Modules vs. Templates

"Module input (or output) signature does not match. Select updated input (or output) parameters."

This warning can appear if you are using cloned versions of the pre-packaged callback templates (see Shared Modules). In most cases, this indicates that the input or output parameters of the cloned module do not match the parameters that Designer expects to find in the original template.

This type of mismatch can occur if the original template has since been updated with a new parameter that doesn't exist in the cloned module. It can also occur if you are using a module that was cloned in an earlier version of Designer. For example, you might be using a module in Designer 9 that was cloned from a template in Designer 8.

To resolve this warning, you can try the following:

- Check the original template to compare the input and output parameters with the ones in the cloned module.
  - Go to Shared Modules > Templates and open the callback template that corresponds to the cloned module.
  - In the Initialize phase of the template, check the User Variables for any input and output parameters that are missing from the cloned module.
  - Add the missing parameters to the cloned module.
  - Go to the application that contains the Callback V2 block and re-publish the application.
**Important**

If you see this warning for an application that is currently in Live production, it will not have any effect on how callbacks are processed in your Live production environment. Callbacks continue to be processed normally.

**Publishing failure due to parseInt() function**

If an application fails to publish due to validation errors with the `parseInt` function, go to the block with the error and check if the function contains a second parameter for the optional `radix` value. In most cases, Designer displays a validation warning if this value is missing, but does not prevent the application from being published. However, if an application does fail to publish due to an error with this function, try adding the `radix` value.

Example:

```javascript
parseInt(, 10)
```

After making this change, save your changes and try the publish operation again.
Contents

• 1 Introduction
• 2 How to get started
  • 2.1 Review the package components
  • 2.2 Configure and publish the data tables
  • 2.3 Publish the applications
• 3 Applications
  • 3.1 Engage Cloud Email
  • 3.2 Send Engage cloud Email
  • 3.3 Send Engage cloud Email v2
• 4 Modules
  • 4.1 Email HOOPS Check
  • 4.2 Email Send Automated Message
  • 4.3 Email Routing
• 5 Data Tables
  • 5.1 Email Profiles
  • 5.2 Email HOOPS Service
  • 5.3 Routing Service
• 6 Message Collection
  • 6.1 Email
• 7 Known issues and limitations
Introduction

Designer includes a pre-built digital application package that integrates Designer interaction handling and routing capabilities with the Genesys Multicloud CX Email functionality provided by Intelligent Workload Distribution (IWD) and Agent Workspace.

The package enables you to:

- Process incoming emails from Workload Manager and route them to the intended agents.
- Send reply emails from agents to customers (with support for Quality Assurance reviews).

For more information about Genesys Multicloud CX Email, see How email works.

How to get started

Review the package components

First, review this page to familiarize yourself with all of the package components. The package includes the following Designer objects and resources (each links to the section where they are described in more detail):

- Applications (digital type) for routing and sending emails. These applications are designed to be ready for you to publish and use.
- Modules to perform specific functions within the applications, such as checking the hours of operation, sending automated messages, and determining which routing services to use.
- Data Tables to manage the configuration settings that are specific to your operations. You'll use these to define email profiles, hours of operation, and routing criteria.
- A digital Message Collection you can use when sending auto-acknowledgement messages to customers.

Configure and publish the data tables

Next, you'll need to update the package data tables with the required configuration settings for your
site. Remember to publish the data tables after you are finished making your changes.

Publish the applications

Finally, publish the email applications.

Important

Some of the package contents in your deployment might include the term IWD in their names. For example, the Send Email application might appear as IWD Send Email. They are, however, the same resources as described on this page.

Applications

The package includes three pre-built digital applications that are designed to be ready-to-use. Rather than make changes to the applications directly, you can use the data tables and various business controls resources to customize the applications for your requirements. For example, you can use the data tables to create profiles that tell the application how to handle emails for certain categories, which business hours to check, or which skill expressions to target.

The package includes the following applications:

Engage Cloud Email

This is the main application that handles the processing and routing of incoming emails from Engage Cloud Email.

Here's a quick summary of the application flow:

- When an email arrives, the application checks which category the email belongs to and matches it to the appropriate Email Profile. If there is no matching profile, the default profile is used. (Emails that are determined to be automated replies are ignored.)
- The hours of operation are checked. If the contact center is closed, the email is held in a parking queue until the contact center opens. At that point, the application resumes processing the email.
- The email is routed according to its priority. If the last agent who serviced the customer is not available (or the Last Called Agent option is not enabled), the email is routed to a skill expression.
- The agent completes their handling of the email.

Each major event in the application flow adds a milestone to the reporting details that can be used to track the progress of each interaction.
Send Engage cloud Email

This application sends reply emails from agents to contacts using the Send Email block. It can also flag certain emails for supervisor review and route a percentage of those emails to a supervisor skill expression.

Here's a quick summary of the application flow:

- An agent composes a reply email using Agent Workspace. Depending on your setup, standard responses can be managed in eServices Manager or by using Standard Responses Management in Designer.
- The application checks if the email was reviewed by a supervisor. If yes, or it does not require a supervisor review, it sends the email to the recipient.
- If a certain percentage of random emails are to be sent to a supervisor for review, the application routes these to the target supervisor skill expression.

Send Engage cloud Email v2

This application shares the same basic functionality as the Send Engage cloud Email application, but can also perform additional handling when additional Quality Assurance (QA) reviews for email are required. For more information about email QA reviews, see Email Quality Assurance.

Here's a quick summary of the application flow:

- An agent composes and sends a reply email using Agent Workspace. Depending on your setup, standard responses can be managed in eServices Manager or by using Standard Responses Management in Designer.
- The application checks if the email was reviewed for QA and if the disposition was accepted. If yes, or the email does not require a QA review, it sends the email to the recipient. If the disposition was rejected by the reviewer, the email is sent back to the agent so that they can make the required changes.
- When the agent makes the required changes and re-submits the email, the application directs it to the original QA reviewer for their review and approval.
- If the email is approved, the application sends it to the recipient.
- If a certain percentage of random emails are to be sent to a supervisor for review, the application routes these to the target supervisor skill expression.

Tip

If you do need to make changes to an application flow, it is recommended that you clone the application. This creates a new virtual application that you can then modify as desired, while leaving the original application unchanged.
Modules

The modules are smaller chunks of application code that perform specific common functions, such as looking up information in data tables to obtain configuration settings. The Engage cloud Email package contains modules that check for hours of operation, send automated messages, and determine which email routing service to use.

The package includes the following modules:

Email HOOPS Check

This module checks the Email HOOPS Services data table for the Hours of Operation, such as Business Hours, Emergency Flags, and Special Days. If the contact center is closed, a Route Digital block parks the interaction until business hours resume. At that point, the parked interaction is processed and given a higher priority.

Email Send Automated Message

This module uses an Automated Message Block to send an auto-acknowledgement message to customers confirming that their email was received. The Email message collection contains messages you can use for this purpose.

Email Routing

This module checks the Email Routing data table to determine how the interaction is to be routed.

Data Tables

The data tables included with the package provide the configuration settings that the modules reference while performing their various functions. You can use the data tables to control how emails are handled by creating profiles that each have their own customized settings.

Tip

Remember! If you make changes to a data table, you must publish it for the changes to take effect. If you need more information about working with data tables, see Editing a data table.

The package includes the following data tables:

Email Profiles

Each row in this data table is a profile that controls how the application handles an email. You can think of a profile as being like a virtual application that can be customized to handle emails in a
specific way. For example, you could set up a profile for a particular line-of-business or segment that
aligns with the categories defined in Workload Manager.

The profile ID is initially set to default, but can be changed if you define a new profile. If an email
does not have a category, or there is no corresponding profile found, the default profile is used.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile ID** (primary key)</td>
<td>profile_id</td>
<td>A unique ID that the system automatically assigns to the profile.</td>
</tr>
<tr>
<td>Profile Name</td>
<td>profile_name</td>
<td>You can assign a unique name to the profile so that it appears as a virtual application for reporting purposes. This enables you to differentiate it from any other profiles or instances of the application that are running.</td>
</tr>
<tr>
<td>Profile Description</td>
<td>profile_description</td>
<td>If desired, you can provide a description for the profile. For example, you might associate this profile with a particular line of business or customer segment.</td>
</tr>
<tr>
<td>HOOPS Service</td>
<td>hoops_service_id</td>
<td>The ID of the HOOPS Service defined for this profile in the Email HOOPS Service data table.</td>
</tr>
<tr>
<td>Routing Service</td>
<td>routing_service_id</td>
<td>The ID of the Routing Service defined for this profile in the Email Routing data table.</td>
</tr>
</tbody>
</table>

**Email HOOPS Service**

This data table is where you'll specify the Email HOOPS (Hours of Operations) service definitions. Each service can contain one of each type of Business Controls object, i.e. Business Hours, Emergency Flags, or Special Days.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOOPS Service ID ** (primary key)</td>
<td>hoops_service_id</td>
<td>A unique ID that the system assigns to this service.</td>
</tr>
<tr>
<td>Open Hours</td>
<td>business_hours</td>
<td>Specify the Business Hours object to use for this service.</td>
</tr>
<tr>
<td>Holidays</td>
<td>special_days</td>
<td>Specify the Special Days object to use for this service.</td>
</tr>
</tbody>
</table>

**Routing Service**

This data table is where you'll configure the settings for routing services. You can define multiple routing attempts, and each attempt can route to a different target.
Important
Currently, only skill expressions are supported as routing targets.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Service** (primary key)</td>
<td>routing_service_id</td>
<td>You can enter a unique name for the routing service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: Multiple attempts will use the same routing service if the attempts are related.</td>
</tr>
<tr>
<td>Routing Attempt** (primary key)</td>
<td>routing_attempt</td>
<td>Specify how many routing attempts are to be made. Attempts are made in increasing order.</td>
</tr>
<tr>
<td>Skill Expression</td>
<td>routing_skill</td>
<td>Specify the target skill expression to be used for routing.</td>
</tr>
<tr>
<td>Last Agent Routing</td>
<td>last_agent_routing</td>
<td>(Boolean) If the <strong>Last Called Agent</strong> option is enabled, the email is routed to the agent that last serviced the customer. Otherwise, the email is routed to the specified skill expression.</td>
</tr>
<tr>
<td>Supervisor Review</td>
<td>supervisor_review</td>
<td>(Boolean) If enabled, the email is considered for supervisor review.</td>
</tr>
<tr>
<td>Review Percent</td>
<td>review_percent</td>
<td>The percentage of emails considered for supervisor review that will be routed to the specified supervisor skill expression.</td>
</tr>
<tr>
<td>Supervisor Skill Expression</td>
<td>supervisor_skill</td>
<td>Specify the target supervisor skill expression to use for routing.</td>
</tr>
<tr>
<td>Routing Timeout</td>
<td>routing_timeout</td>
<td>Specify the timeout for routing attempts.</td>
</tr>
</tbody>
</table>

Message Collection

The package includes the following Message Collection:

Email

The **Email** message collection contains messages that can be used for sending auto-acknowledgment emails with the Automated Message Block. You can access this collection from the Digital Resources page.
Known issues and limitations

The following issues are known to affect certain deployments and will be addressed in future releases of Designer:

- Quality Assurance reviewers are not able to accept or reject emails sent to them for review.
- Emails that should go back to an original reviewer for a follow-up review are not reaching the intended target.
SMART Inbound Voice Routing Application

Contents

• 1 Creating a new application profile
• 2 Creating or editing IVR menu options
• 3 Application resources
  • 3.1 Application
  • 3.2 Shared Modules
  • 3.3 Data Tables
  • 3.4 Business Hours
  • 3.5 Special Days
  • 3.6 Media Resources
The SMART Inbound Voice Routing application is a baseline application designed to operate with multiple profiles that you can define and manage using Business Controls.

**Related documentation:**

- 

At a basic level, the SMART Inbound Voice Routing application works as follows:

- A customer calls your contact center.
- The application checks the defined business controls (hours of business, special days, and emergency flags) and plays a corresponding message.
- The customer selects an option from a menu, which the application then uses to route the call to an appropriate agent.

You can use the application "as is", with the default profile, or create and customize additional profiles to offer a unique experience for a particular customer segment or line of business.

The application package includes several Designer object resources, such as shared modules, data tables, and media resources. You can learn more about the resources that are used by the SMART application here.

**Creating a new application profile**

The application profiles are contained in the M1 Profiles data table. While the default profile can be used for many business scenarios, the SMART Inbound Voice Routing application is structured to let you quickly create new application profiles that can be customized for your own business needs.

Keep in mind that the new application profile can only support settings that are available for SMART-type applications. It's best if the new application serves a similar business purpose or requires a similar type of flow as the original baseline application.

To create a new application profile, add a new row to the M1 Profiles data table and configure the settings for each of the columns. To edit the data table, go to Business Controls > Data Tables and select the M1 Profiles data table for editing.

Here are some recommendations and guidelines:

**Profile ID and Profile Name**

These must be unique. The ID could be a line of business (LOB), department name, DNIS, or another unique value that has business significance (for example, don't use the department name as an ID if you are setting up more than one application profile for it). The profile name will be used for reporting purposes, so you should give it a name that distinguishes it from any other profiles.

**HOOPS Service ID**
If the profile shares HOOPS (Hours of Operation) checks with another profile, you can just reuse the same HOOPS row and specify the ID. If not, add a new HOOPS row to the M1 HOOPS Services data table and specify its ID.

**IVR Menu Service ID**
If offering a menu, specify its ID from the M1 IVR Menus table. If not, leave it blank.

**Routing Service ID**
Specify the ID of a Routing Service from the M1 Routing Services table to use if the IVR Menus do not specify a routing service.

Assuming the new application profile uses existing HOOPS, IVR Menus, and Routing Services, it is ready to go. But you will also need to change the application initialization logic to select this newly created profile based on your criteria.

For example, if the profile ID is a DNIS, select the Assign profile to this session or call block and set varMainProfile = DNIS, as shown here:

For reporting purposes, in the Analytics dashboards, the application profiles are associated with the baseline SMART application, but each profile has reporting data associated with its assigned name.

### Creating or editing IVR menu options

To create a new menu service, go to Business Controls > Data Tables and select the M1 IVR Menus data table for editing.

Add a new row and specify the Menu Name, Menu Audio, and Menu Level. Set the other values according to how the menu will be used.

For example, if this menu is to be offered if the caller presses 2 from the first-level menu, you would set it up like this:

- In the first-level menu row, set Menu Options to use the new menu name. Let's say we've called it Leasing.
• In the example below, the **Next Menu** for DTMF option 3 has been set to **Leasing**, which becomes the second-level menu. If a caller presses 3 when the first-level menu is offered, they are offered the **Leasing** menu next.

• For this particular scenario, we don't want to specify a **Routing Service** for the **Leasing** option. If we do, the **M1 IVR Menus** module will end and the application will start routing the call according to the specified service. So it is left blank.

• All related menus must have the same **IVR Service ID**, so make sure that both the first-level menu option and the newly-defined second-level menu option have the same **IVR Service ID** value. Otherwise, they will not be treated as related menu levels.

**Application resources**

The following Designer objects are included as part of the SMART Inbound Voice Routing application package:

**Application**

• **SMART M1**

  This is the main application. The flow has been structured to provide some of the most common functions of a Designer application, such as calling modules to check for various conditions (like holidays or emergencies), offer self-service menus, and route calls to designated targets, while still allowing for some customization.
Shared Modules

Shared modules are smaller application "chunks" that work with their related data tables to perform various functions within the main application, such as checking for certain conditions, offering menus, or routing calls.

- **M1 HOOPS Checks**
  Checks for various conditions that can trigger an early-exit from the application. For example, it checks if there is an emergency flag, a holiday, or if the call is outside the scheduled hours of operation.

- **M1 IVR Menus**
  Presents a series of menu choices to a caller and then routes the call according to how the caller responds.

- **M1 Routing**
  Used for routing a call to an intended target (such as a **Skill Expression, Agent Group, or Direct Number**).

Data Tables

Data tables contain data values that are used by the related modules and other application objects. You can edit the data tables directly to add or make changes to various options.

M1 Profiles

Each row in this data table is a *profile* that influences how the application behaves and processes the incoming call. You can think of a profile as being like a *virtual application*, as each application profile will handle calls in a different way. To add a new application profile, simply add a new row to this table (see Creating a new application profile).

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile ID**</td>
<td>profile_id</td>
<td>Internal ID for uniquely identifying a profile. This can be a string, a VQ name, or the DNIS. (There is a <strong>default</strong> profile already created.)</td>
</tr>
<tr>
<td>Profile Name</td>
<td>profile_name</td>
<td>The application profile name. This name helps to distinguish the various application profiles for reporting purposes.</td>
</tr>
<tr>
<td>Profile Description</td>
<td>profile_description</td>
<td>Description for this profile.</td>
</tr>
<tr>
<td>HOOPS Service ID</td>
<td>hoops_service_id</td>
<td>ID of the HOOPS Service defined in the HOOPS data table.</td>
</tr>
<tr>
<td>Default Menu Service ID</td>
<td>menu_service_id</td>
<td>ID of the IVR Menu Service defined in the IVR Menus data table.</td>
</tr>
<tr>
<td>Default Routing Service ID</td>
<td>routing_service_id</td>
<td>ID of the default Routing Service defined in the M1 Routing data table. This is used if the selected IVR Menus don't define their own</td>
</tr>
</tbody>
</table>
### M1 HOOPS Services

This data table contains early-exit criteria objects, typically related to Business Controls settings. For example, it specifies which message to play when a call is received during an emergency, a holiday, or outside of the defined business hours.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOOPS Service ID**</td>
<td>hoops_service_id</td>
<td>Internal ID for uniquely identifying this service.</td>
</tr>
<tr>
<td>Emergency Flag</td>
<td>emergency_flag</td>
<td>Boolean value that indicates whether or not there is an emergency.</td>
</tr>
<tr>
<td>Emergency Message</td>
<td>emergency_message</td>
<td>Emergency message.</td>
</tr>
<tr>
<td>Open Hours</td>
<td>business_hours</td>
<td>Points to a business hours object.</td>
</tr>
<tr>
<td>Closed Hours Message</td>
<td>business_hours_message</td>
<td>Specifies the message to be played if the current time is outside the defined business hours.</td>
</tr>
<tr>
<td>Holidays</td>
<td>special_days</td>
<td>Points to a special days object.</td>
</tr>
<tr>
<td>Holidays Messages</td>
<td>special_days_message</td>
<td>Message to be played if the current date is a holiday.</td>
</tr>
<tr>
<td>Queue Music</td>
<td>music_in_queue</td>
<td>(Reserved for future use.)</td>
</tr>
</tbody>
</table>

### M1 IVR Menus

This data table contains multiple levels of nested menus that are offered to callers in the Self Service (IVR) phase. Each menu can enable DTMF options 1-9 and take one of the following actions for the selected option:

- Go to another menu, as per the target menu you've specified in the option settings, or
- Start routing the call, as per the routing service you've specified in the option settings.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Column ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVR Service ID**</td>
<td>ivr_service_id</td>
<td>ID for the IVR Menu Service.</td>
</tr>
<tr>
<td>Menu ID**</td>
<td>menu_id</td>
<td>Unique ID of a Menu within a Menu Service.</td>
</tr>
<tr>
<td>Menu Name</td>
<td>menu_name</td>
<td>User-friendly name for the Menu.</td>
</tr>
<tr>
<td>Menu Description</td>
<td>menu_description</td>
<td>Description of what the Menu does.</td>
</tr>
<tr>
<td>Menu Level**</td>
<td>menu_level</td>
<td>Numeric value indicating the nesting level of a Menu. The first (or top) level is 1.</td>
</tr>
</tbody>
</table>
**Display Name** | **Column ID** | **Description**
--- | --- | ---
Menu Audio | menu_audio | Announcement to be played that will offer the Menu.
Menu Options | menu_options | Options and actions for this menu.
Menu Type | menu_type | (Reserved for future use.)

**M1 Routing**

This data table controls routing services, and contains definitions of multiple routing attempts, each of which can route to a different type of target (such as a **Skill Expression**, **Agent Group**, or **Direct Number**).

<table>
<thead>
<tr>
<th><strong>Display Name</strong></th>
<th><strong>Column ID</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Service**</td>
<td>routing_service_id</td>
<td>Unique name for the routing service. Keep in mind that multiple attempts will have the same service if the attempts are related.</td>
</tr>
<tr>
<td>Routing Attempt**</td>
<td>routing_attempt</td>
<td>Integer indicating the attempt number. Attempts are made in increasing order.</td>
</tr>
<tr>
<td>Routing Option</td>
<td>routing_type</td>
<td>Type of routing. This must be one of <strong>Skill</strong>, <strong>Direct Number</strong>, or <strong>Agent Group</strong>. The exact string is required.</td>
</tr>
<tr>
<td>Skill Expression</td>
<td>routing_skill</td>
<td>Skill expression for routing.</td>
</tr>
<tr>
<td>Agent Group</td>
<td>routing_agent_group</td>
<td>Agent Group for routing.</td>
</tr>
<tr>
<td>Direct Number</td>
<td>routing_direct_number</td>
<td>Agent Group for routing.</td>
</tr>
<tr>
<td>Voice Mailbox</td>
<td>routing_voicemail</td>
<td>(Reserved for future use.)</td>
</tr>
<tr>
<td>Virtual Queue</td>
<td>virtual_queue</td>
<td>Virtual Queue for routing.</td>
</tr>
<tr>
<td>Queue Music</td>
<td>music_in_queue</td>
<td>In-queue audio.</td>
</tr>
<tr>
<td>Duration</td>
<td>routing_duration</td>
<td>Duration for routing attempt.</td>
</tr>
</tbody>
</table>

**Business Hours**

- **Main Office Hours**
  The business hours are predefined as Mon-Fri, 9 AM to 5 PM.

**Special Days**

- **Office Holiday**
  The special days are predefined with a list of US holidays.
Media Resources

These include multiple audio files, as contained within the **Shared Audio** collection found under Media Resources.

**Important**

The SMART M1 application does not include any Speech Grammars.
Default Routing in Designer

Contents

• 1 About Default Routing in Designer
• 2 Application resiliency
  • 2.1 External API fetch failures
  • 2.2 Business Controls lookup failures
  • 2.3 Scripting errors
• 3 Platform resiliency
• 4 Default routing
Learn how default routing works in Designer.

**Related documentation:**

**About Default Routing in Designer**

Designer applications provide self service to interactions (voice calls, digital interactions) and route them according to the business logic. These applications may fail to provide these services due to a variety of reasons, which can negatively impact the end user experience and the ability to deliver a standard of interaction processing that aligns with business objectives.

This page describes Default Routing as it pertains to the **voice channel** on Genesys Multicloud CX. Basically, Default Routing acts as a fallback application for routing voice calls to an agent in the event that an application experiences a service failure, to ensure that these interactions are not lost. In many cases, the triggering of Default Routing can be avoided by improving the ability of applications to handle some common scenarios, which is described in Application resiliency.

**Application resiliency**

Designer applications are rarely self-contained. They almost always rely on data that is retrieved dynamically from multiple sources, such as data tables, customer CRM APIs, and others. As such, it is important to be aware that any data lookup is prone to failure and that applications must check for this condition and take it into account before proceeding forward.

A lookup failure may be considered as recoverable if the application can lookup another data source or provide a default value that satisfies the business logic and provides an acceptable level of service. In such cases, the application must test the outcome of each lookup and if a failure occurs, assign these defaults before moving on to subsequent lookups in a chain or with executing any business logic that relies on this lookup.

Other failures may be considered critical where it may not be possible to continue to provide the desired level of service, but even these cases should be handled by the application with an action such as routing to a route point or global skill.

Common failures that may be encountered by applications are described below. These cases must be handled in the Designer application to ensure these are handled correctly and do not trigger Default Routing.
External API fetch failures

APIs are called from the HTTP REST and Custom Service blocks. Both report on the outcome using the output properties in the **Results** tab. This outcome must be checked to confirm the API call succeeded before using its output. Failures can occur due to remote API issues, load, latency, or incorrect configuration.

---

**Properties - HTTP REST**

This block is used to fetch data from HTTP REST based services

![Service details](image)

- **Service details**
- **Authentication**
- **Results**
- **Advanced**
- **Test**

This variable will be set to true if the fetch operation is successful, and false otherwise

- `varLookupError`

The data from the HTTP response will be stored in this variable

- Choose variable

The headers from the HTTP response will be stored in this variable (as JSON with header names being the keys)

- Choose variable

If the fetch returns with an HTTP error, the error code will be stored in this variable

- Choose variable

Error handling, perform this action if the fetch operation is not successful

- Please choose

---

Business Controls lookup failures

Business Controls (Data Tables, Business Hours, Special Days, and Emergency flags) are almost always external to applications. Successfully reading, or evaluating them in application logic, is not guaranteed. All Business Controls blocks (with the exception of Emergency) provide the status of the lookup using the **Result** tab variables in addition to output variables that provide the actual result the application logic is interested in.

In this example, the variable `varLookupError` will be set to `true` if the operation of looking up a data table encounters an error. In that case, `varRoutingSkillsFromTable` will not contain usable data, and referencing this variable from a routing block will generate an error.
Scripting errors

Designer uses a flavor of ECMAScript/JavaScript that can be embedded in blocks such as Assign Variables, Segmentation, and others. These scripts may have robustness issues that could lead to runtime errors when the script is evaluated. Certain block properties, such as Advanced Scripting in the Assign Variables block, will catch these errors and provide an outcome variable similar to Business Controls blocks to signal that there was a problematic evaluation of the script.

However, other block properties have stricter requirements for scripting expressions, and will jump to Finalize if the application runs and encounters an issue with the expression. These expressions are normally much simpler than scripts entered in Advanced Scripting tab of Assign Variables blocks, but it is still essential to ensure they are robust. For example, if a data table lookup fails, and the next block assumes that the lookup was successful and tries to operate on the returned data, it will run into an error and cause unintended consequences.

Important

Note that lack of proper error handling in External Services (API) and Business Controls blocks can show up as scripting errors. If an Assign Variables block relies on a data table result and tries to extract certain properties from it without proper guard conditions, this will generate a script error if the data table lookup fails and returns no data.
Platform resiliency

Genesys Multicloud CX is a highly reliable and scalable platform. However, it is important to put mitigation in place for Designer applications for the unlikely event that the platform experiences technical issues and cannot reliably run these applications. This obviously falls into the critical category and will rely on Default Routing to continue routing the call to a global skill or designated route point.

Default routing

Default Routing relies on a different, simple, and standalone application. It is deliberately kept simple to make it more resilient. This application may be another Designer application which can be edited and managed from Designer, or it can be an IRD System application that contains multiple enhancements to handle a number of platform failure scenarios, as well as failures in Designer application logic.

Here’s a brief comparison of these two application types:

<table>
<thead>
<tr>
<th>Designer Application</th>
<th>IRD System Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed using Designer.</td>
<td>Cannot be managed by users of Designer.</td>
</tr>
<tr>
<td>Can support business logic, such as LOB separation.</td>
<td>Generic handling. Application is kept simple for robustness. May not support LOBs/departments/divisions.</td>
</tr>
<tr>
<td>Will be triggered when applications experience errors.</td>
<td>Will be triggered when applications experience errors.</td>
</tr>
<tr>
<td>May be affected by platform issues that affect Designer applications.</td>
<td>More likely to be resilient if platform issues affect Designer applications.</td>
</tr>
</tbody>
</table>

Important

Setting up Default Routing

Default routing cannot be set up directly in Designer and must be performed by Genesys. To learn more, contact your Genesys Representative.

In addition to handling application errors, Default Routing also detects if there is no audio playback from the Designer application during an active call and will initiate after 10 seconds of silence. This limit is not configurable and the application logic must ensure audio is played at all times and that silences are kept to a minimum.

Here are some suggestions for maintaining audio playback to avoid prolonged periods of silence:

- Play a prompt as soon as possible at the beginning of the application. Keep processing in the Initialize phase limited to under 2 seconds, ideally less than 1 second.
- In Self Service, play a generic prompt first, and only then proceed to business controls checks.
• Do not daisy-chain multiple lookups without inserting some audio in between.

• Keep external API call timeouts less than 10 seconds and enable Play fetch audio, which plays specified audio during the entire lookup.

**Important**

**Terminating Calls skip Default Routing**

It may be obvious, but if the Designer application terminates the call, Default Routing will not be triggered. If the application logic decides certain lookup failures are critical or it has encountered another critical errors, it can simply jump to Finalize using a GoTo block. This allows Default Routing to be initiated after a certain timeout.