Genesys Designer Help

Shared Modules

4/12/2026

Shared Modules

Contents

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

There are four types of shared modules:

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

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

Using Shared Modules

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

Shared Module blocks

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

Routing blocks (as "Busy Treatments")

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

application is triggered.

- Learn more about routing blocks and busy treatments

Start Treatment block

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

- Learn more about the Start Treatment block and busy treatments

Creating a Shared Module

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

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

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

Importing a Shared Module

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

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

Settings

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

General Tab

Managing Audio Resources for Shared Modules

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

Example

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

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

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

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

Milestone Path Prefix

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

DTMF Options Tab

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

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

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

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

Global DTMF Options Among Shared Modules

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

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

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

Other Actions

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

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

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

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

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

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

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

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