Genesys Designer Help

Application Phases

4/18/2025

Application Phases

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

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

Initialize

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

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

Self Service

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

phase - **Assisted Service**.

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

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

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

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

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

Assisted Service

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

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

The following are typical actions for this phase:

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

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

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

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

Finalize

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

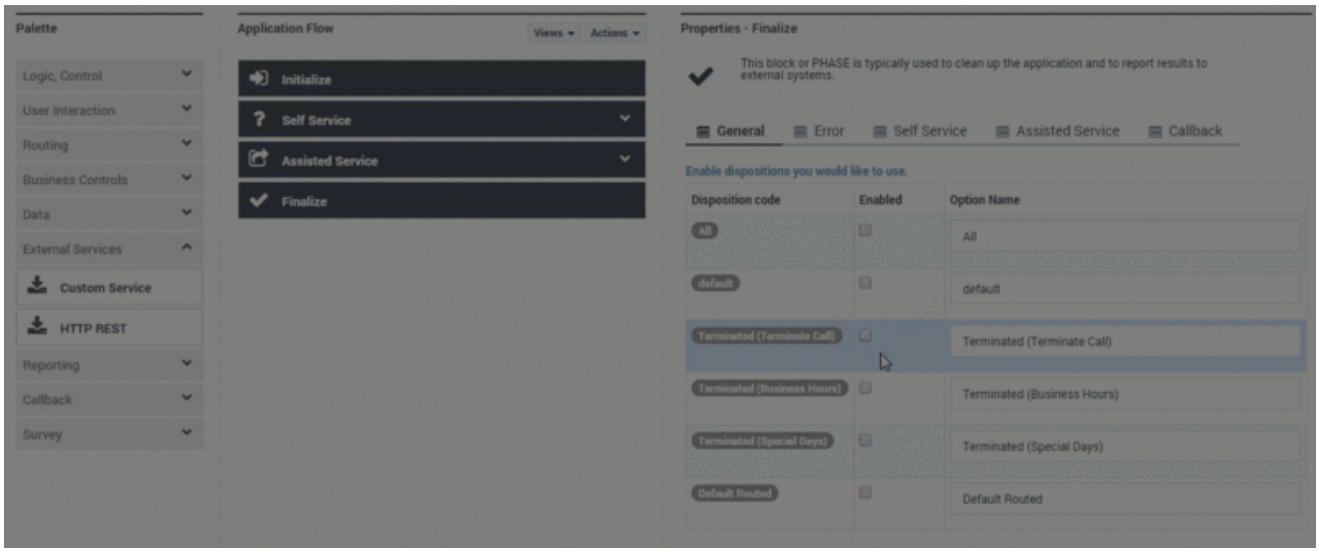
The following are examples of typical scenarios:

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

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

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

Here's an example (click to enlarge):



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

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

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