Genesys Predictive Routing Deployment and Operations Guide

Deploy the URS Strategy Subroutines

5/10/2024

Contents

Genesys Predictive Routing (GPR) provides subroutines components that are integrated with your Genesys Routing solution. The subroutines are placed within an existing strategy, where they add agent scoring and best-match functionality that enables you to fine-tune the routing of a specific interaction to the agent who can best handle it, based on the KPIs you want to optimize.

Related documentation:

-

About the URS Strategy Subroutines component

This topic explains how to use the preconfigured strategy subroutines with Interaction Routing Designer (IRD) and Universal Routing Server (URS).

NOTE: The information in this topic applies to environments running GPR URS Strategy Subroutines 9.0.016.00 or later, Universal Routing Server (URS) 8.1.400.60 or later, and Interaction Routing Designer 8.1.400.39 or later. If you are running older versions of these components, see Deploy the URS Strategy Subroutines in the previous documentation set.

This topic includes the following information:

- List of subroutines and other files included in the IP, with brief explanations of what they do.
- Deployment procedures.
- See Routing Scenarios Using GPR for a discussion of how the subroutines handle agent-interactions matching in various scenarios.
- For a description of the Designer routing block for Predictive Routing and how to configure it, see Predictive Routing Block.

> **Important**
>
> If you would like to evaluate Genesys Predictive Routing for use with schedule-based routing (using Genesys Workforce Management), service-level routing, or business-objective routing, contact Genesys Professional Services for a review of your routing environment.

The following is a high-level overview of the steps required to deploy the URS Strategy Subroutines:

1. Configure URS to support Predictive Routing.
2. Import the subroutines.

3. Review the list of subroutines and other files included in the IP, with brief explanations of what they do.
4. Define the entry points in your IRD strategy for the appropriate subroutines.
5. Set appropriate values for the strategy subroutine configuration options, which are located in the Predictive_Route_DataCfg `Transaction List` object.
6. Configure the parameters for the subroutines used in your environment.
7. Test that the subroutines are correctly directing interactions to agents.

NOTE: Genesys does not support custom subroutines for Predictive Routing in a hybrid environment.

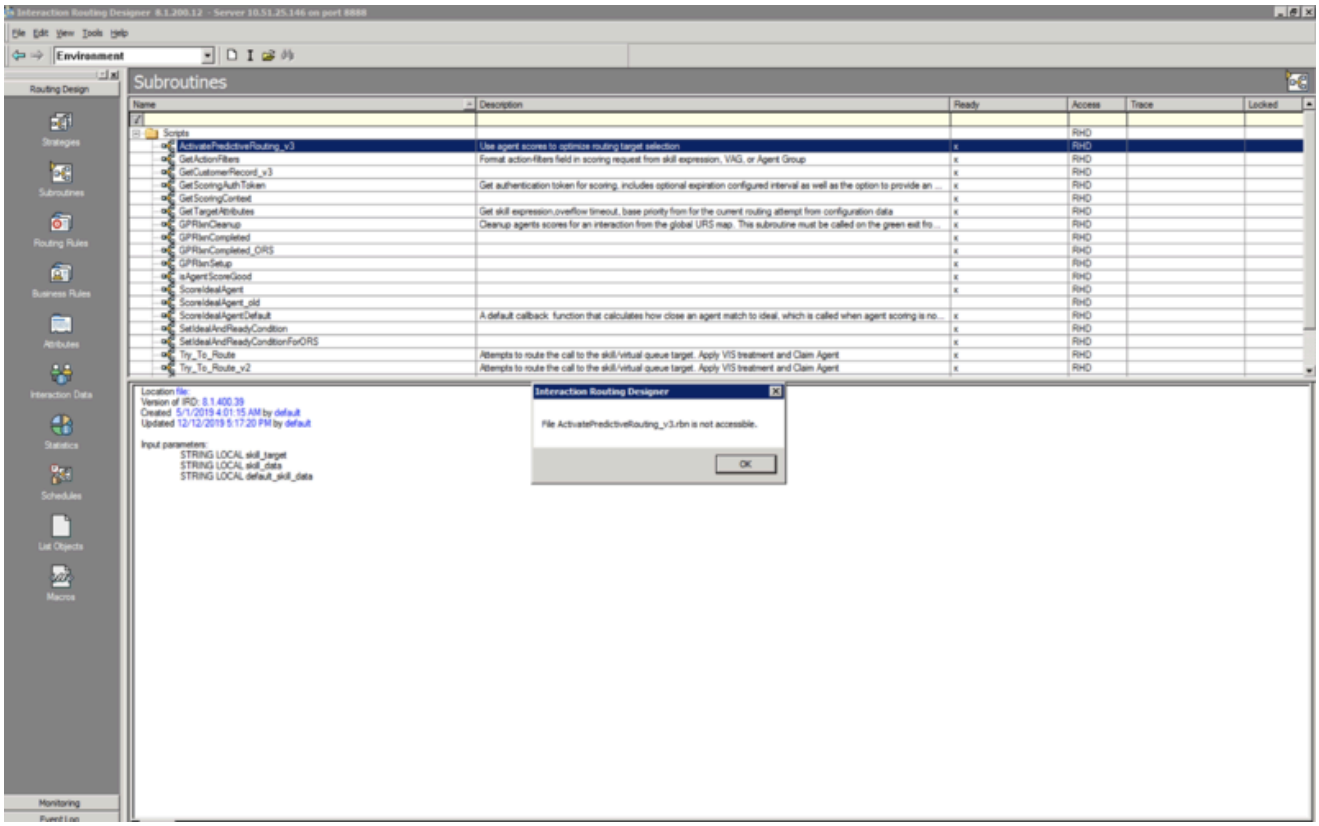
Install the subroutines

To download and import the GPR Strategy Subroutines into your routing environment, perform the following steps:

1. Copy the installation package to the host where Interaction Routing Designer (IRD) is installed.
2. Open a command prompt and navigate to the folder where you put the installation package folder.
3. Run the `install.bat` command. This command starts the import script, which prompts you for the following input parameters:
   - Full path of the IRD installation directory.
   - Configuration Server host name.
   - Configuration Server port number.
   - User name to connect to Configuration Server. This username must have WRITE access to Configuration Server.
   - Password to connect to Configuration Server.
   - IRD `Application` object name.

     After you enter the required parameters, the script imports all the subroutines, statistics, macros, and the `Transaction List` object into Configuration Server. The result of the import process is generated in the **/subroutines/ImportResult** folder.
4. Open IRD and verify that the GPR Strategy Subroutines are present. The correct input and output parameters are specified by default.

NOTE: When you select a subroutine to view its properties, the Location file property is empty. If you try to open one of the GPR subroutines, an error message appears stating that the RBN file is not accessible (as shown in the image below). *This is normal expected behavior*. It means that the import was successful and you can now invoke the GPR subroutines from the routing strategy with the required inputs.



What is included in the installation package

The IP for the URS Strategy Subroutines component contains the following files and folders:

- **install.bat** - A helper script to import the subroutines into your environment.
- The **subroutines** folder, which contains the following ZCF subroutines files:
  - **ActivatePredictiveRouting_v3.zcf** - Called from a routing strategy when the conditions you specify are met. This subroutine automatically calls additional subroutines that score agents, rank the scored agents, and evaluate whether there is an agent available to handle the interaction based on agent score. Referred to just as *ActivatePredictiveRouting* in the remainder of this topic.
  - **GetActionFilters.zcf** - GPR in a hybrid environment supports filtering performed by URS only. You must set the ***use-action-filters*** option to **false**. In this configuration, GPR identifies the list of agents matching the target skill group along with the configured login status expression. This information is also reported in the action filters of the scoring request.
  - **GetScoringAuthToken.zcf** - Called from the ActivatePredictiveRouting subroutine to authenticate with the GPR Core Services Platform.
  - **GPRIxnCompleted.zcf** - Computes the values for the GPR KVPs based on the selected agent scores and updates the URS global map.
  - **isAgentScoreGood.zcf** - URS calls this subroutine to suppress routing to an agent who is in ready state if this agent does not provide an acceptable match for the interaction. You can use this

subroutine in conjunction with relaxation thresholds to target better-matched agents preferentially, expanding the pool of agents if the best-matched agents are unavailable.

- **SetIdealAndReadyCondition.zcf** - Called from ActivatePredictiveRouting subroutine. This is a wrapper subroutine around scheduling the calls to subroutines such as ScoreIdealAgent and isAgentScoreGood, which are executed outside the main interaction processing flow. Implements the GPR operation modes controlled by the ***prr-mode*** option.
- **ScoreIdealAgent.zcf** - Called by URS before routing an interaction to an Agent Group or Virtual Agent Group (VAG) at the time URS invokes the SelectDN function. This subroutine sorts the agents within the target group according to their scores, in decreasing order. It can also rescale the agent score to the range accepted by URS, if necessary.
- **GetScoringContext.zcf** - Looks for the customer feature key names used by the Predictor in URS global map, if already stored. If none is found, this subroutine makes an API call to GPR and fetches the Predictor details. From the returned details, it parses the customer_features_schema field and stores the necessary key names in URS global map with the expiration value set to 86400 seconds (one day).
- All user data matching the predictor feature keys and containing nonempty values is sent in the scoring request, except for any keys specifically excluded using the ***udata-keys-to-exclude*** option.
- **GPRIxnSetup.zcf** - Creates the interaction global map in URS memory with the ConnectionID as the key name; adds the Genesys Info Mart KVPs and initializes them with the default values; sets the gpmMode parameter to `off` and gpmResult to 15 (Predictive Routing is turned off or not used for this interaction). See Configure GPRIxnSetup for complete configuration instructions and how to use this subroutine.
- **GPRIxnCleanup.zcf** - Sends outcome and other scoring data to the score log and attaches them to EventUserEvent in the form of KVPs for storage in the Genesys Info Mart database. The attached UserEvent data includes details required for Predictive Routing performance analysis, such as gpmStatus, which indicates whether, when the interaction was routed, there was an agent-surplus or an interaction-surplus situation. Also cancels the execution of the ScoreIdealAgent and isAgentScoreGood callback subroutines if an attempt to route an interaction with Predictive Routing times out. The strategy then tries to route the interaction using skill-based routing.
- This subroutine requires you to configure certain parameters. See Configure GPRIxnCleanup for instructions.
- This subroutine must be called from the default port of the last Routing block used for Predictive Routing. You must also set the **Clear targets** flag in that Routing block. Alternatively, you can configure the strategy to loop back to the same Routing block again after calling the GPRIxnCleanup subroutine.

- **objects.kvlt** - Contains the following statistics and macros, and the `Transaction List` object used to configure the GPR subroutines in your Genesys Configuration environment. The objects included in this file are the following:
  - **Print_Log_Message** macro, which is used for printing messages in GPR subroutines.
  - **RStatGPRAgentsReadyOrACWvoice** - Custom statistic for identifying agents who are in the Ready or the ACW state. (introduced in 9.0.016.00)
  - **RStatAgentsTotal**
  - **RStatAgentsReadyVoice**
  - **StatAgentOccupancy**
  - The template file to create the **Predictive_Route_DataCfg** Transaction List object, with the required configuration options and default values.

NOTE: Carefully review and verify the default values set in the **Predictive_Route_DataCfg** `Transaction List` object to ensure that they are appropriate for your environment.

Subroutines for Environments Using Dynamic Priority Routing

The URS Strategy Subroutines provide the following subroutines that provide improved dynamic-priority interaction handling:

- **ActivatePredictiveRouting_v3** - This subroutine does the following:
  1. The main routing strategy should set the initial priority for the interaction. The Initialize Variables block takes the initial priority parameter from the skill data `List` object and saves it to the parBasePriority local variable.
  2. After a scoring request is completed and the agent scores are placed into the URS Global Map linked to the interaction, the subroutine executes priority increments. This happens once per interaction.
  3. The subroutine verifies whether the ***set-dynamic-priority*** option is set to `true`. The following steps are executed only on the first routing attempt when the option is set to `true`.
  4. The subroutine reads the remaining priority options configured on the Predictive_Route_DataCfg `Transaction List` object: (***priority-increment***, ***priority-init-interval***, and ***priority-interval***).
  5. The subroutine calls the IncrementPriorityEx function with the required arguments.

Subroutines for Environments Using Non-ASCII Encoding

The URS Strategy Subroutines support non-ASCII encoding (by default, all GPR components use UTF-8 encoding). The subroutines specified below include enhancements for non-ASCII environments:

- **ActivatePredictiveRouting_v3** - Converts non-ASCII characters to UTF-8 characters before sending scoring requests to the Predictive Routing scoring engine.
- **GPRIxnCleanup** - Converts non-ASCII characters to UTF-8 characters before sending scoring outcome data to the score logs.
- **GetScoringAuthToken** - Replaces the StrFormat function with the SetStringKey function. This eliminates an issue with the *~s* character in the StrFormat function, which does not work correctly in a non-ASCII environment.

Subroutines for WFM schedule-based routing

Release 9.0.018.01 and higher of the GPR URS Strategy Subroutines support an integration with WFM schedule-based routing, also known as WFM activity-based routing. When you enable WFM schedule-based routing, WFM Server provides a target set of agents to handle each interaction. The following subroutines are enhanced to integrate GPR with WFM schedule-based routing:

- **ActivatePredictiveRouting_v3** - This subroutine has the following enhancements:
  - When used in conjunction with the WFM schedule-based routing, the subroutine expects the following parameters to be added to the skill_data input list:
    - **use_wfm_schedule** = `true`
    - **activity_name** = name of WFM Activity for the current interaction>
    - **is_wfm_agent_empty** = `true/false`

When these parameters are added to the skill_data field, the subroutine expects the input to the skill_target parameter to be a list of agents produced by the ExpandWFMActivity[] function. The main strategy invokes this function prior to calling ActivatePredictiveRouting_v3.

  - If the **use_wfm_schedule** flag is not included in the data passed to the skill_data field, the input for the skill_target is expected to be a virtual agent group (VAG) string, which can be a skill expression or an Agent Group name, or a combination of skill expression and Agent Group name.
  - The expected format for the list of agents that the main strategy passes to the skill_target input parameter is:
    `.A,.A,...,.A`

- If the ExpandWFMActivity[] function produces an empty output, Genesys recommends that the main strategy should capture this error and select a different target, which must be provided to the subroutine as a VAG string. In this scenario, the strategy sets the **use_wfm_schedule** flag to `false` or omits the flag entirely.
  - If you do not configure the main strategy to handle an empty list of agents produced by ExpandWFMActivity[], the main strategy must pass the parameters **is_wfm_agent_empty** = `true` and **use_wfm_schedule** = `true` to the subroutine in the skill_data input variable. The subroutine then reports **gpmResult** = 17 error and **gpmMessage** = `Empty WFM target list` and exits, passing the interaction to skill-based (non-GPR) routing. For more information, see GPR KVPs for Genesys Reporting.
- **GetActionFilters** - This subroutine has the following enhancements:
  - When the **use_wfm_schedule** flag is set to `true` and the value for the **parVAG** input parameter of this subroutine is an agent list (that is, the **is_wfm_agent_empty** parameter is set to `false`), the ActivatePredictiveRouting subroutine sets the **isAgentList** parameter to `true` and passes it to GetActionFilters. If the input to the **parVAG** input parameter is a VAG string, the **isAgentList** parameter is set to `false`.
  - The format of the output is (note that the suffix *.A is removed):
    `,,...,`
- **GPRIxnCompleted** - This subroutine has the following enhancements:
  - When the **use_wfm_schedule** flag is set to `true` and the input for the **parVAG** input parameter is an agent list (that is, the **is_wfm_agent_empty** parameter is set to `false`), the value for the gpmStatus KVP is calculated based on the WFM activity name, which is passed as the **activity_name** input parameter to the ActivatePredictiveRouting_v3 subroutine.

About Workforce Management (WFM)

WFM enables contact center managers to better manage their workforce with accurate staffing plans that take into account the following:

- Projected contact volumes
- Projected average handle times
- The various skills and skill levels of the agent population
- Agents' schedule preferences
- Mandated vacation and break requirements for the agents' locations

See the following documentation sets to to learn more about WFM:

- WFM for Genesys Multicloud CX

- WFM for Genesys Engage on-premises

Configure URS to run Predictive Routing

Perform the following steps to configure URS to work with GPR:

Set the recommended values for the following options

1. In the **[default]** section:
   - **automatic_ideal_agent** - Sets a universal score for all interactions not routed using GPR. This universal score enables you to compare GPR and non-GPR interactions when they are placed into a queue and provides a mechanism for resolution of agent reservation request collisions between GPR and non-GPR interactions processed by different URS nodes.
     - **automatic_ideal_agent** can take a Boolean value (true/false), but for use with GPR, set the value to a number, which is interpreted as the default deviation from the ideal score. GPR uses this value unless it is overwritten by the SetIdealAgent subroutine.
     - Genesys recommends that you set the value of **automatic_ideal_agent** to (-).
     - Use of **automatic_ideal_agent** requires URS version 8.1.400.60 or later.
   - **run_verbose** - Set this option value to `0` for production environments. (In non-production environments only, you can use the value 1.)
   - **static_strategy** - Set this option value to the string `empty`.
   - **vqtime** - Set this option value to `13:2048`.
2. In the **[web]** section:
   - **verbose** - To configure the http log for URS to check scoring requests and responses, set the option value to 3.
   - **http_port** - Set this option value either to `2071` or `web`.
   - **def_trusted_ca** - Specify the certificate authority to use for secure connection if Web Service object does not provide any.
3. In the **[http]** section:
   - **http_port** - Set this option value either to `2072` or `http`.
   - **verbose** - Set the option value to 3.

Enable HTTPS support

No changes to the Subroutines components are required. However, HTTPS support on Unix-based operating systems requires that you install the Genesys Security Pack on the host running Universal Routing Server (URS). See Installing Genesys Security Pack in the *Genesys Security Deployment Guide* for more information.

- For instructions, and a general discussion of HTTPS and TLS support among Genesys components, see the *Genesys Security Deployment Guide*.
- For HTTP/S configuration for URS, see the section "Web Service Connections Using HTTP Bridge" in the

Genesys 8.1 Universal Routing Deployment Guide, the sections on Web Service Option (p. 687), IRD Web Service Object (p.771), and TLS (p. 782) in the Universal Routing Reference Manual, and HTTP Bridge Updates in the Supplement to the *Universal Routing Reference Manual*.

- Only simple TLS is supported. For simple TLS, you need to configure only the URS **def_trusted_ca** option.
- If you are using a chain of trusted certificates (intermediate ca and root ca, in pem file format), you can concatenate them, as described in Configuring Multiple Trusted CAs in the *Genesys Security Deployment Guide*.

Configure the subroutines

To use the strategy subroutines provided with Predictive Routing, perform the following steps:

1. Open the strategy you plan to use with Predictive Routing and place the ActivatePredictiveRouting Subroutine object in the desired location. It must be inserted into the routing strategy before the call to a Routing block or a call to the SelectDN function. The graphic below shows the subroutine insertion points.
   This subroutine requires the following three input parameters:

   - `skill_target:` A STRING indicating the target selected by the URS for an interaction. This can be a virtual agent group or an agent list from WFM.
   - `skill_data:` A LIST, which must contain the following two keys:
     - `overflow_timeout:` A STRING defining a period of time in seconds during which URS tries to route the interaction to the current target.
     - `base_priority:` An INTEGER defining the priority value the interaction has before the call to the ActivatePredictiveRouting subroutine.

       When used in conjunction with the WFM schedule based routing, the ActivatePredictiveRouting subroutine expects the following parameters to be added to the skill_data input list:

     - `use_wfm_schedule = true`
     - `activity_name` = name of WFM Activity for the current interaction>
     - `is_wfm_agent_empty = true/false`
   - `default_skill_data:` A LIST, which must contain the following keys:
     - `AgentScore:` Takes either `Y` or `N` as its value. To activate Predictive Routing, you must set this parameter to `Y`.
     - `predictor:` Contains the name of the section in the Predictive_Route_DataCfg `Transactions List` configuration object that defines predictor configuration.
     - `START_TS:` See gpmWaitTime Configuration for details.) The UTC time indicating when an interaction arrived in the queue.

2. The isAgentScoreGood subroutine checks for interactions that have been in queue for an unusually long time using the `varQueueTimeSec` parameter, which is set to `600` seconds by default.
   - If you are expecting conditions that might result in longer wait times, you might need to set a larger value for this variable. To change the value of this variable, edit the value for the ***setreadycondition-timeout***.

- URS might experience high CPU loads if the timeout is extended beyond 600 seconds and you are using agent hold-out (that is, you have set the value for the ***use-setreadycondition*** option to `true`).

3. The GPRIxnCompleted subroutine can be inserted either as a Custom Routing step in the Routing object or immediately in front of the object in your strategy that contains a call to the RouteCall function.

Set Values for Configuration Options

Set values for the configuration options in the Predictive_Route_DataCfg Transaction List Object.

- **NOTE:** Carefully review and verify the default values set in the **Predictive_Route_DataCfg** `Transaction List` object to ensure that they are appropriate for your environment.
- Among others, the GPR subroutines send the scoring requests to the URL configured in the ***platform-base-url*** option and logging requests to the URL configured in the ***platform-logging-url*** option.
- The ***orig-connid-key*** option is required in all deployments to store the original connection ID, used as a unique identifier, for each interaction.

Configure GPRIxnSetup

The GPRIxnSetup subroutine does the following:

- Creates the interaction global map in URS memory with the ConnectionID (ConnID) as the key name.
- Adds all the Genesys Info Mart KVPs used in GPR and initializes them with the default values. All the KVP values initialized by the GPRIxnSetup are replaced with actual values when the remaining GPR subroutines are invoked. If the GPR subroutines are not invoked, the score log and the Genesys Info Mart database continue to store the default values set in the GPRIxnSetup subroutine.
- Sets the gpmMode parameter to `off` and gpmResult to 15 (Predictive Routing is turned off or not used for the current interaction). This establishes the initial condition in the contact center and establishes clearly when GPR actively begins to score agents and determine routing targets. As soon as interactions are routed using the GPR subroutines, these KVP values are updated to reflect actual contact center conditions.

To report non-GPR calls in the score log and the Genesys Info Mart Database, set the value of the URS **prestrategy** option to `GPRIxnSetup`.

- If you already specify a subroutine in the **prestrategy** option, you can copy the contents of the GPRIxnSetup subroutine into your existing subroutine.
- For a complete description of the **prestrategy** option, see the Universal Routing Reference Manual.

Configure GPRIxnCleanup

The GPRIxnCleanup subroutine must be called from the default port of the last Routing block used for Predictive Routing. This subroutine correctly identifies abandoned interactions and interactions in which GPR was unable to route the interaction and add this information to the score log and the Genesys Info Mart gpmResult KVP. The KVP values used are the following:

- gpmResult = 13, gpmMessage = Call Abandoned
- gpmResult = 14, gpmMessage = Call Routing Failed

GPR does not report on abandoned calls by default. To enable this functionality, configure the GPRIxnCleanup subroutine as follows:

1. Add OnCallAbandoned['GPRIxnCleanup'] in the strategy that invokes the GPR subroutines. If you already use the OnCallAbandoned function to point to another strategy, update the target strategy to call the GPRIxnCleanup subroutine.
2. When routing succeeds, call GPRIxnCleanup with the parameter `true` from the **default** port of the last Routing block used for Predictive Routing.
3. When routing fails, call GPRIxnCleanup with the parameter `false` from the **red** port of the last Routing block used for Predictive Routing.
4. When an interaction is abandoned, GPRIxnCleanup is called automatically with an empty parameter.

Configure Reporting on Both GPR and Non-GPR Interactions

There are a number of scenarios in which interactions routed using GPR and those routed using alternative methods (non-GPR) might target the same pool of agents:

- Non-GPR interactions might be sent to the same queue as GPR interactions.
- A strategy might send some interactions to GPR and to non-GPR routing, based on specified conditions.
- A different strategy might be routing interactions to the same target agents as the GPR interactions.

URS Strategy Subroutines provides support for tracking interactions not routed using GPR. The following KVP values provide the necessary information:

- gpmMode = `off`
- gpmResult = 13 (abandoned) or 14 (routing attempt failed)

> ### Important
> If routing fails for a non-GPR interaction, or it is abandoned, GPR does not record these status conditions. For GPR the following is recorded: gpmResult = 15, gpmMode = `off`, gpmMessage = `Predictive Routing is turned off or not used for this interaction`.

To configure your routing environment to handle both GPR and non-GPR routed interactions, perform the following steps:

1. Configure the gprIxnSetup subroutine, which is require to report correctly on blended GPR and non-GPR routing.
2. Call the following subroutines in all strategies routing interactions to agents also targeted from GPR. You can invoke them directly, without going through the ActivatePredictiveRouting_v3 subroutine.
   - GPRIxnCompleted should be called as a custom routing step in all Target Selection blocks or as a Function block before routing interactions to agents using a RouteCall block.
   - GPRIxnCleanup should be called after all Target Selection blocks or RouteCall blocks.

gpmWaitTime Configuration

The URS Strategy Subroutines use the value of the START_TS variable to calculate when the interaction arrived in the queue. START_TS is a mandatory variable passed in the default_skill_data List object and stored as a KVP for use in Genesys Info Mart reporting.

- gpmWaitTime is calculated based on the difference between: (the UTC time when the agent is selected) - (START_TS variable value).
- The START_TS variable is also used for measuring periods of time in A/B tests.
- In Genesys Info Mart reporting, the value of the START_TS variable is converted to the DateTimeKey function, used to populate the START_DATE_TIME_KEY column in the GPM_FACT table.

Turn off Predictive Routing

You might choose to turn predictive routing off temporarily for A/B testing or troubleshooting. To do so, use one of the following methods:

- Set the prr-mode configuration option to `off`.
- Set AgentScore = `N` in the default_skill_data object parameter passed to the ActivatePredictiveRouting_v3 subroutine.

Upgrade to a new Subroutines release

This upgrade procedure applies to all URS Strategy Subroutines upgrades, including upgrades from a pre-9.0.016.00 release of the URS Strategy Subroutines (that is, from RBN subroutines files to ZCF subroutines files).

NOTE: If you are using custom subroutines, contact your Genesys representative before upgrading.

1. From Genesys Administrator, back up the **Annex** tab of your Predictive_Route_DataCfg `Transaction List` object to an **.arch** file. This preserves your options settings. (The **.arch** file is comparable to a **.cfg** file, but it is required for correct storage of the **anon-salt** value when using anonymization on premise.)
2. If you have customized any of the statistics installed with GPR, make a backup of the customized statistics.
3. Run the import procedure using the **install.bat** installation script, located in the installation package. See Deploy the URS Strategy Subroutines (above) for detailed instructions.
4. After you have imported the new subroutines, return to Genesys Administrator and import your saved configuration options.
   Note: There should be a prompt whether to overwrite the existing data. Choose `no` and then proceed with the import. If there are differences between your current option values and the default values, the import process presents both values so you can choose which to use.
5. If you are using customized statistics, restore them from the backup you made in Step 2.

How subroutines anonymize data

The following subroutines handle data anonymization:

ActivatePredictiveRouting_v3

- This subroutine reads the ***anon-salt***, ***anon-agent-id***, and ***anon-customer-id*** options and stores the values in the interaction variables.
- If the **anon-salt** option has a value configured and **anon-customer-id** is set to `true`, the value in the **context_id** field sent in score requests is anonymized.
- If **anon-salt** has a value configured and **anon-agent-id** is set to `true`, then, having received the scoring response, the subroutine matches actual agent IDs stored in the interaction global map to the hashed agent IDs in the scoring response. All returned scores are stored with the actual agent ID in the global map for further use in the ScoreIdealAgent and isAgentScoreGood subroutines.

GetActionFilters

- If **anon-salt** is has a value configured and **anon-agent-id** is set to `true`, the subroutine sends the anonymized version of the value in the Agent ID field in the score request.
- The subroutine stores the hashed agent IDs together with the actual agent IDs in the interaction-level URS global map so that the ActivatePredictiveRouting_v3 subroutine can map the hashed Agent IDs received in the scoring response with the actual Agent IDs for routing.

GPRIxnCleanup

- If the **anon-salt** option has a value configured and **anon-customer-id** is set to `true`, the subroutine sends the anonymized version of the gpmAgentID value to the score log.
- If the **anon-salt** option has a value configured and **anon-customer-id** is set to `true`, the subroutine sends the anonymized version of the context_id value to the score log.