Genesys Predictive Routing Deployment and Operations Guide

4/12/2026

Table of Contents

Contents

This guide covers the following topics, enabling you to plan, set up, and maintain the on-premises components in your Genesys Predictive Routing (GPR) environment. GPR enables you to match interactions with agents for optimal outcomes.

Related documentation:

-

What's In This Guide

- Quick Start: to get new users up and running
- **Planning:** system requirements, architecture and security, sizing, setting up data for import
- Configuring and installing on-premises components:
  - Deploy Data Loader
  - Configure Data Loader to upload data
  - Configuration Options
  - Start and stop the on-premises components
- **Data Loader logging**
- **Genesys Routing integration**: deploying and optimizing the GPR subroutines
- **Genesys Reporting integration**: configuring GPR to work with the Genesys Reporting components
- The Genesys Predictive Routing Overview video presents a high-level picture of what Predictive Routing can do for you:

Link to video

- Conceptual topics go in-depth on certain aspects of GPR functionality:
  - How does GPR score agents?
  - Routing scenarios using GPR

Looking for Something Else?

Consult the following Help and the Release Notes for other GPR topics:

- Predictive Routing Help shows how to use the GPR application and gives in-depth explanations of the following GPR functionality:
  - Managing accounts, users, roles, and passwords.

    - Viewing your uploaded data
    - Creating training, testing and evaluating predictors and models
    - Users with STAFF-level permissions can access Predictive Routing Staff Help that documents the additional functionality available to them.
- Predictive Routing Release Note for Genesys Multicloud CX - updates and corrections in the GPR Core Platform, which provides the GPR application and the GPR API, and performs agent scoring.
- Data Loader Release Note
- URS Strategy Subroutines Release Note

Quick Start

Contents

To use Genesys Predictive Routing (GPR), you'll need to install and configure the products and components listed below.

Related documentation:

-

Install

1. Install any Genesys components that aren't already part of your environment. You'll need:
   - Genesys Framework
   - Genesys Administrator Extension
   - Universal Routing Server and Interaction Routing Designer
2. Interaction Concentrator
3. Genesys Info Mart
4. Install Genesys Predictive Routing, which consists of the following components:
   - Data Loader—Imports agent and interaction data automatically from the Genesys Info Mart Database. Uploads customer and outcome data from user-prepared CSV files.
     - Data Loader is deployed in a Docker container and requires that you deploy a supported version of Docker.
   - URS Strategy Subroutines

Configure

To complete your setup of Predictive Routing, configure the following components:

1. Set the desired values for the configuration options, as described in the configuration instructions for the URS Strategy Subroutines component and Configure Data Loader to upload data. The complete list of all Predictive Routing configuration options is available from the Genesys Predictive Routing Options Reference.
   You use configuration options to configure a wide range of application behavior, including:
   - The mode GPR is running in, which might be `off`.
   - Schemas and other configuration for Data Loader and data uploads.
   - Login parameters and access URLs.
   - KPI criteria to decide what makes for a better match.

- Scoring thresholds, agent hold-out, and dynamic interaction priority.
- Many other important functions.

2. (STAFF users only) To configure how the match between interactions and agents is determined, follow the procedures accessed from the Create and Train Predictors and Models page, located in the *Predictive Routing Help*.

Review your data

Data Loader draws interaction and agent data from the Genesys Info Mart Database.

Using Data Loader, you can also import agent, customer, and outcome data compiled in .csv format. The GPR web application enables you to view and analyze your uploaded data.

The resulting data is uploaded to the GPR Core Platform as datasets. The agent and customer datasets are given the particular names of *Agent Profile* and *Customer Profile*. A *dataset* is a collection of raw event data. The primary purpose of a dataset is to be the source of predictor data.

- You configure the dataset schemas and upload parameters using configuration options set on the Data Loader `Application` object.
- After Data Loader imports a dataset, you can append additional data as long as it is consistent with the schema that has already been established.

Create predictors and models

Note: Only STAFF users can create predictors and models.

Predictors are based on the dataset information that you have imported.

- A *predictor* defines a view on that underlying dataset. It can select from some or all of the data in the dataset; you can use a predictor with multiple datasets.
- A model is based on a predictor, and uses the same target metric or KVP as that predictor. You can configure multiple models for each predictor. These models can use different selections of the features available in the underlying dataset. Models are the objects actually used to perform agent scoring and interaction matchups.

As you configure a predictor, you can choose which metric you want to work with, what kinds of situations you want to evaluate, and other parameters, constructing a way to determine the Next Best Action in the specified situation, based on the possible actions available at that time. As you gather more data, you can add that new data to your dataset, and have the predictor test against the actual results coming in, enabling you to refine how successful your predictor is.

Reporting and analysis

You can report on various parameters, such as:

- The success of your predictors.
- The results of A/B testing.
- The factors affecting a KPI you are trying to influence.

Reports are available through the following reporting applications:

- The Predictive Routing interface. See Monitor trends and performance in the *Predictive Routing Help* for specific information.
- Genesys CX Insights (GCXI), as part of the Genesys historical reporting offering. The following reports are available in the *Historical Reporting with Genesys CX Insights* documentation:
  - Predictive Routing - AHT & Queue Dashboard
  - Predictive Routing - Model Efficiency Dashboard
  - Predictive Routing Agent Occupancy Dashboard
  - Predictive Routing A/B Testing Report
  - Predictive Routing Detail Report
  - Predictive Routing Operational Report
  - Predictive Routing Queue Statistics Report
- Pulse, which provides the Agent Group KPIs by Predictive Model and Queue KPIs by Predictive Model templates for real-time reporting.

System requirements and interoperability

Contents

Genesys Predictive Routing (GPR) includes several components. This topic provides an overview of the prerequisite hardware and software required to run each component.

It also includes an interoperability table, showing which versions of the Genesys components required to run an end-to-end GPR solution are compatible.

> **Important**
>
> In addition to the prerequisites noted here, see the *Genesys Supported Operating Environment Reference Guide*, which provides operating system, database, and browser requirements information for most Genesys products.

Related documentation:

-

The GPR components: hardware and software requirements

Data Loader

- Connects to Configuration Server to read the Data Loader Application object, which includes the configuration for your datasets.
- Connects to the Genesys Info Mart Database to upload interaction data; enables upload of customer and outcome data from .csv files.
  - Data Loader is deployed in a Docker container. See environment prerequisites and supported versions for your Docker deployment for specific requirements.
  - To enable Data Loader to connect to the GPR Core Platform, you must create the Data Loader user in your account and assign it the SERVICE role. For instructions, see Create the Data Loader user in the *Predictive Routing Help*.
  - To achieve HA, Data Loader uses a primary/backup HA architecture. Failover is controlled by Local Control Agent, which is deployed in the Docker container with Data Loader. See Deploy Data Loader for details.
- Data Loader must be deployed on a host that has no other instances of Local Control Agent installed.

URS Strategy Subroutines

- Out-of-the-box strategy subroutines to use with your Genesys routing components. Genesys Predictive Routing includes a set of subroutines created for use with Universal Routing Server (URS) and Interaction Routing Designer (IRD).

Genesys Reporting Integration

- Configure GPR and your historical reporting components to ensure that the required data, in the form of key-value pairs, is made available to support Genesys historical reporting on GPR performance and outcomes.

Important Considerations Related to Docker

- You might need an active internet connection to download additional libraries when installing Docker.
- The GPR uses the RHEL8 universal base image as the base Docker image.
- To operate correctly in a Docker environment, SELinux (Security Enhanced Linux) should be disabled or running in permissive mode. For instructions, see How to disable SELinux on the Linux web site.
- If you are deploying components in Docker containers in an HA architecture, the system clocks on all target servers must be synchronized. You can use Network Time Protocol (NTP) for this.

Supported Browsers

- Chrome - Latest and one previous
- Internet Explorer - IE 11

System requirements and required components/versions

The following table lists the hardware and software requirements that should be in place before starting your deployment.

Data Loader		
Hardware/Software Type	**Requirement**	**Comments**
OS	• Red Hat Enterprise Linux Server release 8	
Docker	docker-ce version 18.09.2 or higher; OR docker-ee 18.09.2 or higher	
RAM	16 GB	
Configuration Server	8.1.300.36 or higher	
Message Server—for logging	8.1.300.11 or higher	
URS Strategy Subroutines		
Hardware/Software Type	**Requirement**	**Comments**
See the Genesys Predictive Routing Sizing Worksheet to calculate the memory and CPU requirements for URS when using Predictive Routing.		

Interoperability

Among GPR components:

GPR Component	Requirement	Comments
Data Loader	Requires GPR Hybrid environment	

For Routing using the URS Strategy Subroutines:

{! Interaction Routing Designer

Hardware/Software Type	Requirement	Comments
Universal Routing Server	8.1.400.60 or higher	Requires a connection to Stat Server 8.5.108.18 or higher
8.1.400.39 or higher		

For integration with Genesys Reporting:

Hardware/Software Type	Requirement	Comments
Genesys Predictive Routing	9.0.016 or higher	
Interaction Concentrator	8.1.5 or higher	
Genesys Info Mart	8.5.014.19 or higher	
Reporting and Analytics Aggregates	8.5.010.01 or higher	
Genesys CX Insights (GCXI)	9.0.012.01 or higher	

}}

Architecture and security

Contents

This topic presents Genesys Predictive Routing (GPR) architecture, first at a high-level overview, followed by more detailed views of the connections used by Data Loader and the URS Strategy Subroutines with the GPR Platform, which is deployed in the Genesys Multicloud CX.

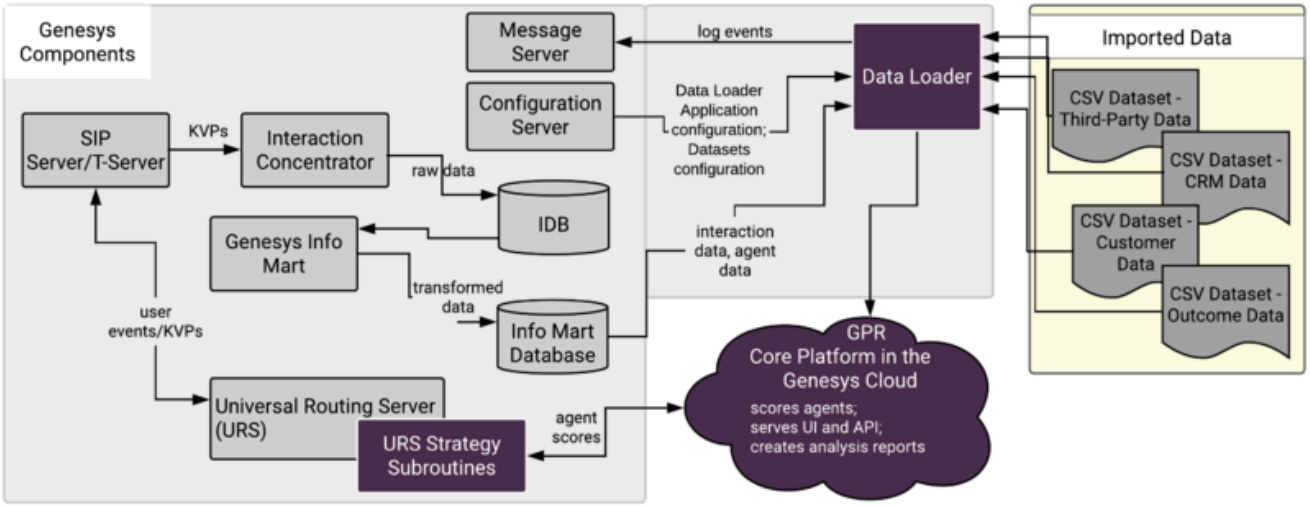
Related documentation:

-

GPR architecture

The following diagram shows a high-level view GPR, how it connects with other Genesys components, and how data enters GPR.

> **Important**
>
> GPR architecture in a high availability (HA) environment is similar to that presented in the diagram except that for HA Data Loader is deployed in a warm-standby primary-backup architecture.



Data Loader architecture

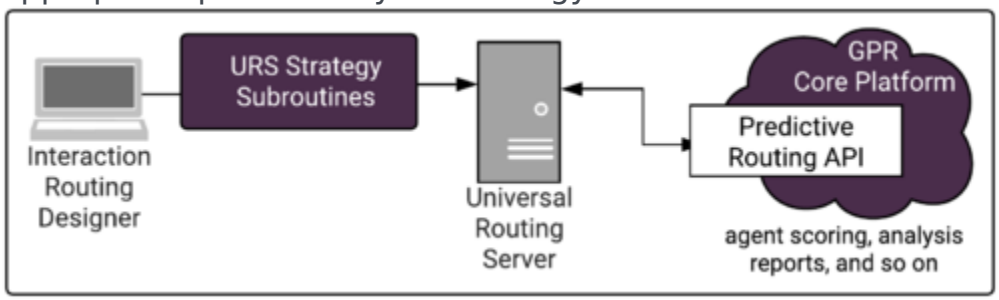
Data Loader is a Java application deployed in a Docker container that you can start, stop, and monitor in Solution Control Interface or Genesys Administrator Extension (GAX). It requires that you deploy

the AI Data Loader Scripts component as well. Data Loader has the following major functionality:

- Connects to the Genesys Info Mart Database for agent-related data, which is uploaded and stored in the Agent Profile. This data includes agent details, such as a location, languages, skills and skill levels, and so on. To configure the schema for this agent data, and to control when Data Loader refreshes data, you must use configuration options in the **[dataset-agent-gim]** section of the Data Loader Application object.
- Connects to Configuration Server to read configuration information for datasets and the Agent and Customer Profiles, as well as other values provided in configuration options, such as the URL to connect to the GPR Core Platform.
  - Data Loader requires a connection to Configuration Server. It does not support connections to Configuration Server Proxy.
- Automatically extracts data from the Genesys Info Mart Database to create datasets.
- For complete configuration information of the Data Loader data upload functionality, see Configure Data Loader to upload data.
- For information on how to create the Data Loader user and assign it the SERVICE role, which enables it to connect to the GPR Core Platform, see Create the Data Loader user in the *Predictive Routing Help*.
- For high availability (HA), uses a primary-backup pair of Data Loader instances, each deployed in a Docker container and monitored through Solution Control Interface (SCI).

Subroutines architecture

Predictive Routing supplies out-of-the-box subroutines for environments running Interaction Routing Designer (IRD) + Universal Routing Server (URS). To deploy the strategy subroutines component, insert the strategy subroutines into the appropriate position in your strategy flow.



> **Important**
>
> Predictive Routing is not supported for environments that use schedule-based routing.

The Subroutines invoke Predictive Routing in real time, sending requests to the GPR Platform, which performs the scoring based on the information you configured in your Predictor and the Model or Models based on it and returns the projected scores for each agent in the target group, indicating

how well they would be expected to handle the specific interaction in question given the particular interaction type, customer intent, agent skill level, and whatever other factors you anticipate to be relevant. URS then chooses the optimal routing target.

Security

Data Loader is delivered in Docker images. This ensures consistent environments from development to production as Docker containers maintain all configurations and dependencies internally, without depending on software installed on host server. With Docker, upgrades are easier and more predictable. Scaling across multiple hosts requires starting the same Docker containers on multiple host servers. In addition, Docker provides isolation; every part of GPR can be scaled separately and has guaranteed access to hardware resources.

Genesys uses the following best practices when it comes to security:

- GPR supports TLS 1.2.
- GPR uses a CentOS 7 Docker image as the base image.
  - SELinux (Security Enhanced Linux) should be disabled or running in permissive mode. For instructions, see How to disable SELinux on the Linux web site.
- The Data Loader Docker image containing Genesys software is continuously scanned for vulnerabilities as part of the build and test pipelines.
- The Data Loader Docker container runs in unprivileged mode.
- Data Loader delivered in a Docker container does not require any additional ports to be open.
- Inside the Docker container, GPR software is executed as a non-root user.

The measures listed above, combined with properly secured host servers, ensures that GPR deployed using Docker containers is as secure as a deployment using more traditional methods.

To understand how Docker containers comply with various security regulations and best practices, see the following pages on the Docker site:

- Docker standards and compliance.
- Docker Security

Secure connections

Predictive Routing supports the following security and connection protocols for Data in Transit:

- ADDP
- HTTPS
- Transport Layer Security (TLS) 1.2

The following protocols are supported for the specified connections:

- Data Loader to Configuration Server: TLS 1.2; you can specify an upgrade-mode Configuration Server

port by updating the **-port** command line parameter in the Data Loader `Application` object **Start Info** tab.

- Data Loader to the GPR Platform: HTTPS
- URS to the GPR Platform: HTTPS

Configure GPR to use HTTPS

GPR supports HTTPS by default.

- To configure HTTPS support for Data Loader, specify HTTPS in the URL Data Loader uses to access the GPR Core Platform. This URS is set in the ***platform-base-url*** option.
- For subroutines-specific configuration, see Configure URS Strategy Subroutines to Use HTTPS.

HTTPS configuration for other components in your Genesys environment is covered in the *Genesys Security Deployment Guide* and in the product-specific documentation.

Secure data

GPR enables you to specify certain data fields as sensitive or personally identifiable information (PII). Data Loader then anonymizes the PII data before uploading it to the cloud for the GPR Core Platform to use.

For more information, see Data anonymization.

Architecture and security FAQs

Q: Is there any effect on on-premises components when the GPR Core Platform, running in the Genesys Multicloud CX, is upgraded?

A: Upgrades are done as rolling upgrades, so there is no downtime. In most cases, there is backwards compatibility with all supported releases of the on-premises components. If any upgrades are required, there will be proper communication about deprecated features and changes to the software that require changes or upgrades to installed components.

Q: How often does Genesys plan to upgrade the Core Platform?

A: As often as required to fix bugs and performance problems or deliver security patches and new features.

Q: How is user authentication handled for the GPR application?

A: The GPR application uses native authentication with a standard username-password combination.

Q: How does GPR handle data anonymization?

A: Every customer/tenant has a unique salt value that is securely encrypted and stored in the ***anon-salt*** Data Loader configuration option. Whenever a new dataset is uploaded, any column you mark as containing sensitive data or personally identifiable information (PII), has all values in that column hashed irreversibly by using sha256 on the tenant_salt + PII value. This hashed outcome is stored in

encrypted form in long-term storage. Anonymization is done purely in memory. PII data is never saved until it has been anonymized. The same process is used during prediction and scoring and the values used for prediction are irreversibly hashed in the same manner.

Q: Does GPR support secure communications?

A: Yes. All communication between on-premises components and the GPR Core Platform is done using TLS 1.2+. All components require HTTPS connections with GPR Core Platform, and support HTTPS Proxy if required.

Q: How do I know whether my on-premises components are compatible and provide support for all the latest features and functionality?

A: Minimum tested interoperable versions are specified in the interoperability matrix.

Sizing worksheet

Contents

Use the Excel worksheet linked below to calculate hardware sizing guidelines for the on-premises GPR components in your environment.

Related documentation:

-

How to fill out the Sizing worksheet

The worksheet has three user input fields (filled in yellow):

- Inbound Call Rate in seconds (Interaction Rate)
- Average Handling Time (in seconds)
- Average Waiting Time in Queue (in seconds)

Macros in the document then automatically calculate the sizing requirements for your environment.

If the URS CPU% is calculated to exceed 80%, the spreadsheet displays an error message.

Click here for the worksheet: Genesys Predictive Routing sizing worksheet

Configure Data Loader to upload data

Contents

Learn how to configure Data Loader to upload data to your instance of the GPR Core Platform for analysis and agent scoring.

Related documentation:

-

Overview

Data Loader uploads data to the GPR Core Platform for analysis and agent scoring. GPR uses the following types of data:

- Agent Profile data and interaction data, which Data Loader uploads using a direct connection to Genesys Info Mart. Once you have configured the connection parameters, Data Loader automatically uploads this data and refreshes it periodically.
- Customer Profile data, which can come from several different sources, depending on your environment. To upload customer data, you must collect it into a .csv file with a consistent schema for Data Loader to upload it.
- Optionally, you can upload more agent data, interaction outcome data, and any other data you find useful. Compile this data into .csv files for Data Loader to upload it.

To upload data, you must create, and specify values for, the configuration options on the **Options** tab of the Data Loader `Application` object. You configure the configuration sections and options in your configuration manager application (GAX or Genesys Administrator).

- The Predictive Routing Options Reference contains a complete list of all Predictive Routing options, with descriptions, default and valid values, and a brief explanation of how to use them.

This topic focuses specifically on data-upload configuration and provides a task-based perspective, enabling you to find which options you must configure for specific upload types and giving more context to help you decide on the appropriate values for your environment.

NOTE: Once you have configured a schema in the Data Loader `Application` object and uploaded your data to the GPR Core Platform, you cannot modify the schema.

Data upload basics

This section explains how to configure Data Loader to upload the essentials for Predictive Routing. These essentials are the Agent Profile dataset, the Customer Profile dataset, and an interaction dataset.

- Create an Agent Profile dataset and a Customer Profile dataset before trying to join interaction, outcome, or other data. The sections that follow cover more in-depth scenarios, including adding other

types of data and joining data.

To upload to Data Loader, open the Data Loader `Application` object in your configuration manager (Genesys Administrator or GAX) and set up the following configuration sections and options for each essential category of data.

1. Agent Profile

To create your Agent Profile dataset, configure the following two sections, and include the options specified within each:

- **dataset-agents-gim**
  - ***sql-query*** - Use the default query provided with Data Loader (the **agents_data_gim.sql** file in the **/dl** folder) or create your own. If you plan to use your own SQL file, be sure to include the mandatory fields listed in Create your own SQL query (below).
  - ***data-type*** - The value for this option must be `agents`.
  - ***upload-dataset*** - This option is set to `true` by default, which tells Data Loader to start uploading data at the interval specified in the ***update-period*** option. To pause uploading, set this option to `false`.
  - ***upload-schedule*** (in release 9.0.018.00 and higher) or ***update-period*** - Specifies how often Data Loader runs an upload process to get new and updated data.
- **schema-agents-gim** - In this section, create one option for each column in your Agent Profile dataset. You must include the columns/options listed below. If you modified your SQL query to read values from additional fields, you must also add those fields as options in this section. The option values are the datatype, and, if the field contains sensitive or PII data, the value anon.
  - **dbID**
  - **EMPLOYEE_ID** (note that this field name must use exactly this format, all caps with an underscore)
  - **groupNames**
  - **skills**
  - **tenantDbID**

2. Customer Profile

To create your Customer Profile dataset, configure the following two sections, and include the options specified within each:

- **dataset-customers**
  - ***csv-separator*** - Tells Data Loader what separator type you are using so it can correctly parse the .csv file.
  - ***data-type*** - The value for this option must be `customers`.
  - ***location*** - Enter the Data Loader Docker container path to the .csv file containing your Customer Profile data.
  - ***upload-dataset*** - This option is set to `true` by default, which tells Data Loader to start uploading data at the interval specified in the ***upload-schedule*** (in release 9.0.018.00 and higher) or ***update-period*** option. To pause uploading, set this option to `false`.

- **schema-customers** - In this section, create one option for each column in your Customer Profile dataset. The option values are the datatype, and, if the field contains sensitive or PII data, the value anon.
  - **CustomerID** - The only required option/column.

3. Interaction Dataset

To create a basic interaction dataset, complete the following steps. In a production environment, you would typically join the interaction data to the Agent and/or Customer Profile datasets. The sections that follow explain how to join data. This section gives you the minimum necessary information to create an initial interaction dataset.

- **dataset-name>** - When you create this configuration section, replace name> with a dataset name of your choice. For example, you can create a section called **dataset-interactions**.
  - ***num-days-upload*** - (In release 9.0.019.01 and higher) Specifies the number of days of data to upload in your initial data upload from Genesys Info Mart.
  - ***chunk-size*** - Defines how many interactions you want Data Loader to upload at once by specifying a period of time. Data Loader uploads all interactions that occur during this amount of time in a single batch.
  - ***upload-schedule*** (in release 9.0.018.00 and higher) or ***update-period*** - Specifies how often Data Loader runs an upload process to get new and updated data.
  - ***sql-query*** - Use the default query provided with Data Loader (the **interaction_data_aht.sql** file in the **/dl** folder) or create your own. If you plan to use your own SQL file, be sure to include the mandatory fields listed in the option description. See Create your own SQL query for instructions.
  - ***data-type*** - The value for this option must be `interactions`.
  - ***upload-dataset*** - This option is set to `true` by default, which tells Data Loader to start uploading data at the interval specified in the ***upload-schedule*** (in release 9.0.018.00 and higher) or ***update-period*** option. To pause uploading, set this option to `false`.
- **schema-name>**- When you create this configuration section, replace name> with the same dataset name you entered for the **dataset-name>** section. If your dataset section is called **dataset-interactions**, this section is **schema-interactions**.

How to join data

Joining data enables you to pull together information about many aspects of your environment, enabling more powerful and nuanced predictions. When you join the data, the interaction dataset is the one containing joined fields. That is, the interaction dataset uploaded to the GPR application displays all its own fields, plus the fields from the joined datasets.

This section explains the basic procedure for joining the Agent Profile and/or Customer Profile dataset with an interaction dataset. The "Advanced upload scenarios" section on this page presents further examples.

Data upload sequence

When joining features from the Agent Profile or Customer Profile datasets to the interactions dataset,

the order in which you upload the datasets initially is important. Upload data in the following sequence:

1. Upload Agent Profile data by setting the **[dataset-agents-gim].upload-dataset** option to `true`. This setting starts upload of Agent Profile data from Genesys Info Mart.
2. After Data Loader starts the agent data upload, configure the Customer Profile data configuration sections and start the Customer Profile data upload by setting the **[dataset-customers].upload-dataset** option to `true`.
3. Then configure the interactions data upload. The procedure below explains how to specify the Agent Profile and Customer Profile dataset fields to be joined to the interaction data. Once this configuration is complete, start the upload of the interaction dataset by setting the **[dataset-name>].upload-dataset** option to `true`.

Data joining procedure

To enable joining, perform the following steps:

1. In the Data Loader `Application` object, create or edit the **dataset-name>** section for your interaction dataset. These instructions refer to the **dataset-interactions** section used as an example in the "Data upload basics" procedures, above. If your dataset has a different name, use your dataset name throughout.
2. Add the ***join*** option. As the option value, specify the names of the datasets to be joined.
   - For example, if you would like to join data from the Agent Profile and Customer Profile datasets with **dataset-interactions**, set the value of the ***join*** option to: **agents-gim**, **customers**, **interactions**.
3. Specify the fields to be joined to the interaction dataset by adding the names of the columns from the joined datasets into the Data Loader **schema-interactions** configuration section.
   - Each joined column must be a separate option in the **schema-interactions** configuration section. The value for each of these options must be the name of the **schema-** section from which the data comes.
   - For example, if you join fields from the Customer Profile (which you created using the **dataset-customers** and **schema-customers** configuration sections) with your interaction dataset, the option value for all joined fields is **schema-customers**.
4. Specify the desired values for the following additional configuration options:
   - ***join-type*** - Configured in the **dataset-interactions** section. Specifies the join type, whether inner (only successfully joined records are uploaded) or outer (all interaction records are uploaded to the platform and the missing data is replaced with `null` values).
   - ***join-keys*** - Configured in the **dataset-** sections of the datasets you are joining to the interaction dataset. Specifies a comma-separated list of the column names by which to join the data from this dataset to the interaction dataset.
   - ***enforce-schema-on-joined-data*** - Specifies whether fields that are not already in the interaction dataset are joined. If set to true, only data from fields already in the interaction dataset are joined. If set to **false**, ALL columns from the joined datasets are added to the interaction dataset. If you set this option to **false**, be careful not to exceed the maximum allowed number of 100 columns in the joined dataset, or Data Loader generates an error and exits the upload process.

The following example shows a **schema-** configuration section for an interaction dataset with joins to the Agent and Customer Profile datasets and an outcome dataset called **feedback**:

[schema-interactions-joined]

- **AHT**=numeric
- **Customer_location**=schema-customers
- **CUSTOMER HOLD DURATION**=numeric
- **CUSTOMER TALK DURATION**=numeric
- **CUSTOMER_ACW_DURATION**=numeric
- **Customer_language**=schema-customers
- **Customer_ANI**=schema-customers
- **EMPLOYEE_ID**=schema-agents-gim
- **Feedback**=schema-feedback
- **groupNames**=schema-agents-gim
- **InteractionID**=string
- **Is_Resolved**=schema-feedback
- **skills**=schema-agents-gim
- **START_TS**=timestamp,created_at

Advanced upload scenarios

In addition to the Agent Profile, Customer Profile, and interactions datasets described in the preceding sections, you can upload other data from your environment. This section presents examples showing how to upload data from additional sources.

Add agent data from sources other than Genesys Info Mart

If your business has employee data relevant for Predictive Routing that is not available from Genesys Info Mart, create a .csv file with that data and configure the Data Loader `Application` object with the following sections and options:

- **dataset-agents-name>**
  - ***csv-separator*** - Tells Data Loader what separator type you are using so it can correctly parse the .csv file.
  - ***data-type*** - The value for this option must be `outcomes`.
  - ***location*** - Enter the Data Loader Docker container path to the .csv file containing your Agent Profile data.
  - ***upload-dataset*** - This option is set to `true` by default, which tells Data Loader to start uploading data at the interval specified in the ***update-period*** option. To pause uploading, set this option to `false`.
- **schema-agents-name>** - Shows how to configure the schema for an agent Dataset to be uploaded from a .csv file. It has only one mandatory field, but you must also configure options specifying the data types for all other fields. Add the anon parameter if the field must be anonymized.

- EMPLOYEE_ID=string,id,anon

Upload outcome data

To upload outcome data from a post-interaction survey, use the following configuration. You can adapt this example to create a dataset based on nearly any data source. In this example, we are uploading a dataset called **feedback**.

- **dataset-feedback**
  - ***csv-separator*** - Tells Data Loader what separator type you are using so it can correctly parse the .csv file.
  - ***data-type*** - The value for this option must be `outcomes`.
  - ***location*** - Enter the Data Loader Docker container path to the .csv file containing your Feedback data.
  - ***upload-dataset*** - This option is set to `true` by default, which tells Data Loader to start uploading data at the interval specified in the ***update-period*** option. To pause uploading, set this option to `false`.
- **schema-feedback**
- - Customer_ANI=string,anon
  - EMPLOYEE_ID=string,anon
  - Feedback=string
  - InteractionID=string
  - Is_Resolved=boolean
  - Timestamp_Feedback=timestamp,created_at

Upload aggregated interaction data to your Agent Profile dataset

Note: In Data Loader release 9.0.017.01 and higher, the cloud-based feature engineering pipeline (CFEP) provides more robust aggregation capabilities. See Configure for Feature Engineering for details.

Aggregated features enable you to pull together data from various sources to create features that address specific scenarios. For example, you can define a feature that gives you the average handle time (AHT) for the interactions each agent in the Agent Profile dataset received from a certain virtual queue over a specified period.

To configure aggregated features in the Agent Profile, replace the **dataset-agents-gim** and **schema-agents-gim** configuration sections with the following:

- **dataset-agents-gim-ext** - *In addition to* the standard options configured for the **dataset-agents-gim** section, add the following options:
  - start-date
  - end-date
  - chunk-size

- **schema-agents-gim-ext** - Be sure to add all fields that Data Loader must aggregate as options in this configuration section.

Create your own SQL query

For your reference, the Data Loader container includes the following two default SQL queries:

- **/dl/interaction_data_aht.sql** - the query used to collect average handling time (AHT) data for Data Loader to upload to the interactions dataset.
- **/dl/agents_data_gim.sql** - the query used to collect data to populate the default Agent Profile dataset.

To review these SQL queries, follow these steps:

1. Run the following command to locate the container ID:
   ```
   docker ps
   ```
2. Use the response from Step 1 to run the following command, which accesses the container:
   ```
   docker exec -it /bin/bash
   ```
3. Navigate to the **/dl/** folder in the container and use the following command to view the default SQL query files:
   ```
   cat
   ```

To create your own custom SQL query, use the following procedure:

1. Review the requirements for mandatory fields contained in this section (below) and follow the guidelines to create your queries.
2. Place your custom SQL query files into one of the subfolders in your installation directory. The installation directories are mapped to the container. The folders available on the host computer are the following:
   - **/ai-data-loader-scripts/scripts/datasets_customers**
   - **/ai-data-loader-scripts/scripts/datasets_agents**
   - **/ai-data-loader-scripts/scripts/datasets_outcomes**
3. For each dataset using a custom SQL query, use GAX to open the associated dataset configuration section in the Data Loader **Application** object. Specify the path to the query file in the ***sql-query*** option. The path must be given relative to the container folder structure.
   For a table containing the mapping between folders on the host and in the container, see Mapping between folders on the host and in the container.

Mandatory fields for custom SQL query files

The following fields are mandatory in your agent data SQL file:

- **EMPLOYEE_ID** - The id of the agent. Must match the name of the ID Field in the Agent Profile dataset.
- **dbID** - The agent's configuration database DBID. Obtained from the Genesys Info Mart RESOURCE_.RESOURCE_CFG_DBID field.
- **tenantDbID** - Obtained from the Genesys Info Mart RESOURCE_.tenant_key field.

- **groupNames** - List of group names assigned to the agent.
- **skills** - Dictionary containing the agent's current skills.

The following fields are mandatory in your interaction data SQL file:

- **InteractionID** - Obtained from the Genesys Info Mart INTERACTION_FACT.media_server_ixn_guid field.
- **EMPLOYEE_ID** - field that defines the Agent ID, which must match the name of the id_field in the agents dataset
- **WHERE clause** - Must include the following:
  - irf.start_date_time_key - This value must fall between the dates set in the :start_ts and :end_ts parameters. Data Loader replaces the :start_ts and :end_ts fields with the start and end time of the interactions dataset chunk, in epoch format.
  - **Note:** The abbreviation **irf** in the WHERE clause indicates the INTERACTION_RESOURCE_FACT table in the Genesys Info Mart database.

To create a custom SQL query when using the cloud feature engineering pipeline (CFEP), set the **use-cloud-feature-engineering** option to **true** and create your query using the following SQL template placeholders:

- :user_data_fields in the SELECT clause and :user_data_joins in the WHERE clause to extract the user data stored in the Info Mart database.
- :vq_filter in the WHERE clause to enable filtering of the interaction records by the Virtual Queue names defined in the ***vq-filter*** option.
- If you use multiple mapping rules for one KVP, you need to specify how Data Loader should include them in the dataset.

NOTES:

- The field names are case-sensitive.
- Use of angle brackets to signify "not" is not supported. Genesys recommends that you use **!=** instead.

Configure the data upload schedule

How often Data Loader uploads data from Genesys Info Mart and how much data it uploads at once is controlled by configuration options. The options available depend on your Data Loader version.

- For releases 9.0.018.00 and higher, you can use the ***upload-schedule*** option to configure precise schedules in CRON format.
- For releases before 9.0.017.01, or if you choose not to use the CRON scheduling, two options, ***update-period*** and ***chunk-size***, coordinate to control upload timing and size.

9.0.018.00 and higher

A CRON expression is a string consisting of six or seven subexpressions (fields) that describe individual details of the schedule. These fields, separated by white space, can contain any of the

allowed values with various combinations of the allowed characters for that field. The following table shows the fields in the order expected and the allowed values for each.

Name	Required	Allowed Values	Allowed Special Characters
Seconds	Y	0-59	, - * /
Minutes	Y	0-59	, - * /
Hours	Y	0-23	, - * /
Day of month	Y	1-31	, - * ? / L W C
Month	Y	0-11 or JAN-DEC	, - * /
Day of week	Y	1-7 or SUN-SAT	, - * ? / L C #
Year	N	empty or 1970-2099	, - * /

Example Cron Expressions

Cron expressions can be as simple as * * * * ? * or as complex as 0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2010.

The table contains various Cron expressions and their meanings.

Expression	Means
0 0 12 * * ?	Start dataset upload at 12:00 PM (noon) every day
0 15 10 ? * *	Start dataset upload at 10:15 AM every day
0 15 10 * * ?	Start dataset upload at 10:15 AM every day
0 15 10 * * ? *	Start dataset upload at 10:15 AM every day
0 15 10 * * ? 2005	Start dataset upload at 10:15 AM every day during the year 2005
0 * 14 * * ?	Start dataset upload every minute starting at 2:00 PM and ending at 2:59 PM, every day
0 0/5 14 * * ?	Start dataset upload every 5 minutes starting at 2:00 PM and ending at 2:55 PM, every day
0 0/5 14,18 * * ?	Start dataset upload every 5 minutes starting at 2:00 PM and ending at 2:55 PM, AND fire every 5 minutes starting at 6:00 PM and ending at 6:55 PM, every day
0 0-5 14 * * ?	Start dataset upload every minute starting at 2:00 PM and ending at 2:05 PM, every day
0 10,44 14 ? 3 WED	Start dataset upload at 2:10 PM and at 2:44 PM every Wednesday in the month of March
0 15 10 ? * MON-FRI	Start dataset upload at 10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Start dataset upload at 10:15 AM on the 15th day of every month
0 15 10 L * ?	Start dataset upload at 10:15 AM on the last day of every month
0 15 10 ? * 6L	Start dataset upload at 10:15 AM on the last Friday

Expression	Means
	of every month
0 15 10 ? * 6L	Start dataset upload at 10:15 AM on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Start dataset upload at 10:15 AM on every last Friday of every month during the years 2002, 2003, 2004, and 2005
0 15 10 ? * 6#3	Start dataset upload at 10:15 AM on the third Friday of every month
0 0 12 1/5 * ?	Start dataset upload at 12 PM (noon) every 5 days every month, starting on the first day of the month
0 11 11 11 11 ?	Start dataset upload every November 11 at 11:11 AM

9.0.017.01 and lower

The ***update-period*** option specifies the interval at which Data Loader attempts to upload data, enabling fresh data stored in the Genesys Info Mart database to be automatically uploaded to the associated dataset. Used with **dataset-agents-gim** and the main interactions dataset, which are the datasets created directly from Genesys Info Mart data.

- If the **update-period** value is less than the value for the **chunk-size** option, Data Loader uploads all data after the watermark marking the end of the previous upload.
- If the **update-period** value is larger than the value of the **chunk-size** option, Data Loader uploads all data after the watermark, split into chunks of the size specified by the value of the **chunk-size** option.

Examples

NOTE: In the the examples below the value of the **end-date** option is set in the future.

- If **update-period** is set to 1 day (P1D) and **chunk-size** is set to one hour (PT1H), all the data after the previous watermark is uploaded in 1-hour chunks. This chunking is designed to prevent overloading your infrastructure.
- If you are uploading a dataset for the first time and set **start-date** to 90 days in the past, **update-period** to 1 day (P1D), and **chunk-size** to 30 days, Data Loader uploads the 90 days of data in three 30-day chunks.

Deploy Data Loader on EKS

Contents

- Administrator

Feature coming soon! Deploy Data Loader in Docker containers hosted on Amazon Elastic Kubernetes Service (EKS).

Related documentation:

-

This page describes the process of installing/configuring/starting the Data Loader application in a Kubernetes environment, specifically AWS EKS. The same instructions are applicable to Azure or GCP deployments with appropriate modifications as needed.

Currently only AWS EKS has been tested and certified. Refer to AWS documentation for creating and managing clusters in your environment.

The compatible Data Loader image version for use in EKS is 9.0.020.00 (or higher). This can be requested from Genesys Software Delivery team. You can install the DL image in a private repository such as JFrog for use in the EKS cluster .

Infrastructure Assumptions

- AWS EKS Cluster
- Nodes
  - To run DL pod, the worker nodes should have enough memory and CPU count to match the recommended resource limit for DL pod(Memory: 16 GB & CPU count: 4)
- Networking
  - Config Server, GIM, and GPR should be accessible from the worker node where DL pod is running.
  - DL deployment in EKS will be exposed as service through ELB(Load Balancer).
  - LoadBalancer URL will be used as host in Config Server for the DL application.
- Persistent Storage
  - To run custom SQL files which is used to fetch the data from GIM.
  - To preserve the log files of the DL application.

Set up the Environment

Create a namespace for Data Loader using the following command:

```
kubectl apply -f namespace.json
```

The **namespace.json** defines how the namespace is to be created. Modify the following sample to suit your environment:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "dataloader",
    "labels": {
      "name": "dataloader"
    }
  }
}
```

> **Important**
>
> - Replicas are not required for Data Loader.
> - Ensure that the resources have a 4 CPU count and at least 16 GB memory available.
> - In AWS EKS, namespaces can be used in a multi-tenant environment.

Before using the data loader image, create the secrets for the image using the following command:

```
kubectl create secret docker-registry jfrogcred --docker-server= pureengage-docker-
staging.jfrog.io --docker-username= --docker-password= --docker-email= -n dl
```

Deploy Data Loader

Ensure that the Data Loader image is available in the cluster or pulled from the repo defined in the **deployment.yaml** file.

Create a Config Map file

The **ConfigMap.yaml** file stores the environment variables used by the Data Loader container. These values are read from the **deployment.yaml** file when the deployment is created.

To create the **ConfigMap.yaml** file, run

```
kubectl apply -f DL_ConfigMap.yaml -n dataloader
```

OR

```
kubectl apply -f DL_ConfigMap.yaml --namespace=dataloader
```

A sample **ConfigMap.yaml** file:

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: dl-primary-config
  namespace: dl # Assumption is namespace dl is already created
data:
  JVMPARAMS: "-server -Xmx2g -Xms2g -Xss512k"
  START_DL: "false"
  CONFIG_HOST: "10.52.86.42"
  CONFIG_PORT: "8888"
  CONFIG_DL_APP_NAME: "DataLoader-Alpha"
  LCA_PORT: "4999"
```

Configure Persistent Storage

A persistent storage is required for storing the log files and read SQL scripts for processing CSV files. Persistent Volumes are resources in the cluster and Persistent Volume Claims are requests for those resources and can also act as claim checks for the resource.

The following configuration will map the local volume to the Kubernetes cluster.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: dl-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/dl_data"
```

The following sample will be used during deployment to mount the volume in the cluster.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dl-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

> **Important**
>
> The Persistent Volume Claim will use the persistent volume for writing the logs and reading SQL scripts.

Deploying the Data Loader application

The data Loader application can be deployed using the standard Kubernetes deployment mode and

can be used to control the Data Loader container pod in t he AWS EKS cluster to manage and scale the Data Loader application.

To deploy the Data Loader application using a YAML file, run

```
kubectl apply -f DL.yaml -n dataloader
```

A sample DL.yaml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dataloader-primary
  namespace: dl
  labels:
    app: dataloader-primary
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dataloader-primary
  template:
    metadata:
      labels:
        app: dataloader-primary
    spec:
      volumes: # Assumption is persistent storage claim dl-pv-claim is already created for
storing the sql files and logs
      - name: dl-pv-storage
        persistentVolumeClaim:
          claimName: dl-pv-claim
      imagePullSecrets:
      - name: jfrogcred # Assumption is secret jfrogcred is already created to pull the image
      containers:
      - name: dataloader-primary
        image: pureengage-docker-staging.jfrog.io/gpr/ai-data-loader:9.0.020.00
        resources:
          limits:
            cpu: "4"
            memory: "8Gi"
        volumeMounts:
        - mountPath: /dl/mount # Based on the mount path, dl application should be configured
to write the log files or read the sql files in the mount path.
          name: dl-pv-storage
        ports:
        - containerPort: 8091
        - containerPort: 4999
        env:
        - name: JVMPARAMS
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: JVMPARAMS
        - name: START_DL
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: START_DL
        - name: CONFIG_HOST
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
```

```
            key: CONFIG_HOST
        - name: CONFIG_PORT
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: CONFIG_PORT
        - name: CONFIG_DL_APP_NAME
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: CONFIG_DL_APP_NAME
        - name: LCA_PORT
          valueFrom:
            configMapKeyRef:
              name: dl-primary-config
              key: LCA_PORT
```

Configure High Availabliity

Setting up a load balancer

To set up a load balancer, modify the following configuration in the YAML file,

```
apiVersion: v1
kind: Service
metadata:
  name: dl-primary-service
  namespace: dl
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
    service.beta.kubernetes.io/aws-load-balancer-internal: "true"
spec:
  loadBalancerSourceRanges:
  - 10.52.0.0/17
  - 172.20.0.0/17
  type: LoadBalancer
  selector:
    app: dataloader-primary
  ports:
    - protocol: "TCP"
      name: "lca"
      port: 4999
      targetPort: 4999
    - protocol: "TCP"
      name: "dl"
      port: 8091
      targetPort: 8091
```

> **Important**
>
> Autoscaling is not supported.

Configure Data Loader for Feature Engineering

Contents

The Cloud Feature Engineering Pipeline (CFEP) facilitates computation of aggregated features and complex data joins for more broad-based predictive analysis.

Related documentation:

-

About feature engineering

The CFEP augments your datasets with engineered features and data joins. It automatically performs complex data selection and manipulation steps that otherwise require much more hands-on effort.

NOTE: Although the CFEP is enabled by default, your Genesys GPR account in the Genesys Multicloud CX must have the CFEP configured before you can use it. Contact your Genesys representative for more information.

After you have arranged to have the CFEP configured on the GPR Core Platform, follow the procedures in this topic to have Data Loader upload your data to the pipeline. The CFEP is available in Data Loader release 9.0.017.01 and higher.

With the CFEP enabled, Data Loader does the following:

- Automatically extracts user data from Genesys Info Mart.
- Uploads historical interaction data processed on specified virtual queues.
- Optionally, triggers execution of the CFEP job every upload period after all dataset chunks are uploaded to the GPR Core Platform.

Turn on feature engineering

Enable feature engineering

Feature engineering is enabled by default. However, you can turn off feature engineering separately for any datasets you do not want to have processed by the CFEP. To turn it off, open the Data Loader **Application** object in GAX and set the value of the ***use-cloud-feature-engineering*** configuration option to `false`. By default, this option is set to `true`.

Start feature engineering automatically

To have the CFEP automatically start processing each time Data Loader uploads fresh data, set the ***trigger-pipeline-execution*** option to `true`. If you prefer to start pipeline processing manually, you can trigger it by sending a request to the GPR API.

Configure data extraction from Genesys Info Mart

SQL Query Templates

The fields that Data Loader extracts from the Genesys Info Mart database for upload are defined by an SQL query file. Genesys provides default SQL query template files with the Data Loader IP to populate the interactions and Agent Profile datasets.

Genesys recommends that you use the standard SQL query templates to upload data to the CFEP. If you need custom SQL queries, review the information in Create your own SQL query for mandatory fields and other important guidelines.

Note: If you use the standard SQL query templates, do not configure the ***sql-query*** option for the interactions and Agent Profile datasets.

Upload User Data

Data Loader extracts user data stored in the Genesys Info Mart database, which is expected to follow user data mapping and propagation rules. See User Data Mapping in the *Genesys Info Mart Deployment Guide* for a discussion of this topic.

- If a user data key in the Info Mart database has only one configured rule, Data Loader automatically adds it to the interactions dataset.
- If you have configured multiple mappings for a user data key and have not specified which is the default rule, Data Loader adds each mapped user data value to a different column in the dataset. For example, a key, **CustomData** has two mappings configured, IRF_ROUTE and PARTY. With no default rule configured, Data Loader adds the following two columns to the dataset: **CustomData_IRF_ROUTE** and **CustomData_PARTY**.
- If one of the mapped user data values is more important than the others for predictor creation and model building, you can specify it as the default rule. See Configure user data with multiple mappings for instructions.

Note: If you do not have multiple mappings for user data in the Info Mart database, or if you are satisfied with Data Loader assigning each mapping to a separate column, you do not need to do any additional configuration.

Filter by VQ

To limit the amount data that Data Loader uploads to the CFEP, list the names of the virtual queues (VQs) from which Data Loader should take historical interaction data.

To specify VQ names, list the names of the desired VQs in the ***vq-filter*** option.

Note: The length of the comma-separated list of VQ names should not exceed 4096 characters. Genesys always recommends that you use Genesys Administrator Extension (GAX) to configure options, but this recommendation is especially important of your list of VQ names is longer than 255 characters.

Migrate datasets to FE

If you have already-configured datasets that you would now like the CFEP to process, use the following procedure.

Note: This procedure must be done by a user with the STAFF role.

1. Disable interaction processing using GPR for the time it takes to perform the migration.
2. From Genesys Administrator Extension (GAX), export your current Data Loader configuration and save it as a reference.
3. Delete the existing Agent Profile schema from the GPR web application. See View the Agent Profile schema in the *Predictive Routing Help* for instructions.

Use GAX to update your Data Loader configuration:

1. Set the ***upload-dataset*** and ***use-cloud-feature-engineering*** options to `false` for all the previously-uploaded datasets you plan to migrate.
2. Remove the following configuration sections, if present. They are not used with the CFEP, which performs data aggregation automatically: **[dataset-agents-gim-ext]** and **[schema-agents-gim-ext]**.
3. Your configuration needs to include both the **[dataset-interactions-gim]** and **[schema-interactions-gim]** pair of configuration sections and a new pair of sections that include the same options and the same initial configuration settings.
4. **NOTE:** Data Loader releases prior to 9.0.017.01 did not include **[dataset-interactions-gim]** and **[schema-interactions-gim]** in the default template. If you do not have those sections in your environment, use GAX to import the Data Loader **Application Template** for release 9.0.017.01 or higher, then continue the following procedure.
5. To create the required configuration sections, use GAX to perform the following steps:
    1. Rename the **[dataset-interactions-gim]** and **[schema-interactions-gim]** sections to **[dataset-interactions-gim-temp]** and **[schema-interactions-gim-temp]**. This prevents them from being overwritten in the following step.
    2. In GAX, import the **DataLoader.cfg** template file, which is located in the Data Loader Installation Package. For detailed information about importing **Application** templates, see Bulk Provisioning of Configuration Options in the *Genesys Administrator Extension Help*.
    3. Locate the newly-created **[dataset-interactions-gim]** and **[schema-interactions-gim]** sections, which were provisioned by default when you imported the **.cfg** file. Rename them for use with the CFEP. For example, the new names of these sections might be **[dataset-interactions-fe]** and **[schema-interactions-fe]**. All further configuration for the CFEP should be done using these sections.
    4. Return to your original sections, now named **[dataset-interactions-gim-temp]** and **[schema-interactions-gim-temp]**. Return their names to **[dataset-interactions-gim]** and **[schema-interactions-gim]**. These sections do not require any additional configuration.
6. (Optional) To use multiple SQL queries to extract data from the Genesys Info Mart database into separate interactions datasets, configure additional **[dataset-name>]** and **[schema-name>]** configuration sections, one for each separate SQL query.
    - **NOTE:** In this case you need to configure the **[dataset-name>]**.***sql-query*** option for each additional dataset and provide the correct path to the associated SQL file location.

7. Configure the date range for each dataset to be used with the CFEP by setting the desired values for the **[dataset-name>]**.***start-date*** and **[dataset-name>]**.***end-date*** options.
8. Save the changes to your configuration.

After configuring your datasets in GAX, continue with the following steps:

1. If you have configured multiple User Data Propagation rules for a single user data key in Genesys Info Mart, see the Configure user data rules section on this page.
2. Install Data Loader release 9.0.017.01 or higher, following the instructions to Deploy Data Loader. The version you deploy must support the CFEP.
3. In GAX, set the ***use-cloud-feature-engineering*** option to `true` for the **[dataset-agents-gim]** and **[dataset-interactions-fe]** configuration sections in the Data Loader **Application** object. If you are using additional datasets, make this change in the associated dataset configuration sections as well.
4. Start Data Loader.
5. In GAX, set the ***upload-dataset*** option to `true` for the **[dataset-agents-gim]** and **[dataset-interactions-fe]** configuration sections in the Data Loader **Application** object. If you are using additional datasets, make this change in the associated dataset configuration sections as well. This triggers the initial upload to the CFEP.
6. After the CFEP job is complete, open the GPR web application and review the datasets created. If necessary, refer to View your uploaded data in the *Predictive Routing Help* for help finding and understanding the data displays. You can now use the uploaded data to create predictors and models.
7. After you have verified the quality of the datasets produced by CFEP, created predictors with the new datasets, and trained the new models, use GAX to remove your old datasets from the Data Loader **Application** configuration.
8. Re-enable GPR interaction processing.

Configure user data rules

If you have used Genesys Info Mart user data mapping rules to have certain user data stored in multiple Info Mart database fields, you can choose to specify one of those as the main value for your dataset. To do so, create an option in the **[schema-interactions-*]** section of the Data Loader **Application**. The option name is the user data key name. The value should indicate the datatype and the parameter 'rule:rule_name>'.

Example

1. If the dataset is identified with the suffix "fe", the section name where you should create the option is **[schema-interactions-fe]**.
2. The option name should be the name of the user data key. In this example, the key name is **CustomData**.
3. Set the option value using the following format: datatype, rule:rule_name>. For example, your option value in this example might be: **string,rule:IRF_ROUTE**.

Data Loader adds a column named **CustomData** to the uploaded dataset, which contains the values from the Info Mart database column defined by the mapping rule IRF_ROUTE.

Note: Data from the other mapping rules is retained and added to the dataset, in case it might prove useful. The value for each rule is stored in a separate column named user_data_keyname>_rule_name>. For example, you might have a rule, PARTY, for the CustomData user data key. In that case, Data Loader adds a column named **CustomData_PARTY** containing the values for the other mapping.

Set up data for import

Contents

Certain requirements and limitations apply to the data that you upload to GPR. This topic explains these requirements, and also presents data security and anonymization.

Related documentation:

-

Supported types of data

In general, you need the following types of data:

Interaction data

- Data Loader automatically extracts interaction data from the Genesys Info Mart database to create datasets.

Agent Profile data

- Data Loader automatically extracts agent data from the Genesys Info Mart Database. You can optionally add agent data from other sources by providing a .csv file for Data Loader to upload.

Customer Profile data

- To create the customer profile, create a .csv file and upload it using Data Loader.

Outcome and other data

- To use outcome data, or data any other sort you find to be relevant for predictive routing, such as results of an after-call survey, create a .csv file and upload it using Data Loader.

See Configure Data Loader to upload data for how to configure Data Loader to upload both Genesys Info Mart data and .csv data.

See the relevant portion of the Help for how to use the GPR application to view your uploaded data and append data to existing datasets.

.csv file size requirements

Use the following guidelines to construct .csv files for data uploads:

- Data Loader uploads data in 512-MB chunks. If your dataset is larger than 512 MB, Data Loader

automatically breaks it into chunks for upload.

- The maximum number of columns in a dataset is 100; the maximum number of rows is 2.5 million. If you upload a file with more than 2.5 million rows, Data Loader uploads only the first 2.5 million and discards the remainder.
- The maximum length of a single column name in a .csv file for upload is 127 characters.
- The maximum length of a single column name that Data Loader will anonymize is 120 characters.
- The maximum number of rows in the agent profile is 20 thousand.
- The number of rows in the customer profile is 20 million.
- The maximum number of columns (features) in the agent and customer profile datasets is configured for each account. The default limit is 50 features for each profile dataset. Only a STAFF user can change this value.

If you try to upload more data than the data size limits allow, GPR generates an error and discards the remaining rows.

When you have reached the size limit, GPR does not add records. However, you can update data associated with previously uploaded records (as identified by the Agent or Customer ID). For example, if you have uploaded 20,000 agents, you cannot add any more. But you can upload the same agents with new values, such as skills or location, and GPR makes those updates.

To add records, you must remove some uploaded records using the GPR API `*/purge` endpoints.

.csv data formatting requirements

- When you create the .csv data file for a dataset, agent profile, or customer profile, do not include the following in the column name for the ID_FIELD, the Agent ID, or the Customer ID:
- ID
- _id
- Any variant of the string `ID` that changes only the capitalization.

Using these strings in the column name results in an error when you try to upload your data.

- When you create the .csv data file for a dataset, agent profile, or customer profile, do not include the following reserved names in column names:
  - created_at
  - tenant_id
  - updated_at
  - acl
- In the agent profile, if you are using skill names that include a dot (period) or a space in them, use double quotation mark characters to enclose the skill name. For example, enter a skill named ''fluent spanish.8'' as "fluent spanish.8".
- GPR supports UTF-8 encoding. All responses and returned data arrives in UTF-8 encoding.

- If you use a Microsoft editor to create your .csv file, remove the carriage return (^M) character before uploading. Microsoft editors such as Excel, WordPad, and NotePad automatically insert this character. For tips on removing the character from Excel files, refer to How to remove carriage returns (line breaks) from cells in Excel 2016, 2013, 2010.
- GPR supports only one-dimensional dictionaries, with up to 200 key-value pairs where the key is a string and the value is int, float, or Boolean. GPR does not support nested dictionaries and lists.
- If you have dictionary-type fields that use comma separators, use tab separators for your .csv file.
- Fields of the dictionary (DICT) type are discovered correctly only if the quotes appear as in the following example, with double quotation marks outside a dictionary entry and single quotation marks for the values within it. This requirement applies to DICT fields in all datasets, including the agent and customer profile datasets.
  - "{'vq_1':0.54,'vq_2':6.43}"
- GPR does not support use of angle brackets () to signify "not" in SQL queries. Genesys recommends that you use the following symbols instead: **!=**.

Data size for models and scoring

The following size limits apply to model creation:

- Maximum number of active models per Tenant - 50
- Total cardinality limit for model training: no specific column count; has been tested up to 250 columns.
  - Total cardinality must be less than 2 to the power of 29.
  - *Total Cardinality* = the number of numeric columns plus the sum of the number of unique values across all string columns within a specified dataset.
- Record count limit for model training - not applicable; from a model-training perspective there is virtually no limit on the number of columns. The constraining issue is the possibility of compromising the model quality by ending up with a reduced number of samples for training.
  - The total number of records must be less than 2 to power of 29 (that is, 536870912) divided by total cardinality as defined above.
  - **Example 1:** You must to use ALL of the data for training the model . If the dataset contains 1 million records, the maximum total cardinality is 536 (536870912 divided by 1 million).
  - **Example 2:** You can *undersample* the data for training the model—that is, use fewer than the ideal number of records for training. You might take 10,000 as the total cardinality, but only 53,687 of your total of 1 million records will be used for training. The calculation to determine this is 10,000 * 53,687 = 536870912 (the maximum cardinality).

The following limitation applies to scoring requests:

- Maximum number of agents that can be scored in one scoring request - 1,000.

Data anonymization

PII, or personally identifiable information, and sensitive data, such as passwords, must be hidden when you upload it to the GPR Core platform. To ensure that sensitive data is secured, instruct Data Loader to anonymize the fields containing such data.

After Data Loader anonymizes the fields you identified as PII, it uploads it securely using TLS.

Note the following points about anonymized data in GPR:

- You can anonymize up to 20 fields in each dataset.
- You cannot anonymize fields *after* you have uploaded data.
- Once you have uploaded data with anonymized fields, you cannot de-anonymize them.
- Anonymizing Numeric or Boolean fields changes them to String fields. This change has some effect on how the fields are weighted in the Feature Analysis report and during scoring.
- Each Tenant has its own unique salt for anonymization.

NOTE: If you anonymize a field, you must anonymize it in every dataset in which it appears. For example, if you anonymize a customer phone number in the customer profile, you must also anonymize it in any dataset in which it appears. If there is an inconsistency, GPR cannot correctly map agents and, as a result, cannot build models for them.

GPR uses the following steps to ensure secure data handling:

1. When Data Loader starts up, it generates a unique 64-character salt string that will be used for anonymization. It stores this string in the **anon-salt** option in the **[default]** section on the **Annex** tab of the primary and backup Data Loader `Application` objects and the Predictive_Route_DataCfg `Transaction List` object.
   - When you open these options in GAX, or any other configuration manager application you use, you cannot see the salt value itself. What you see is an obfuscated version of the salt string.
   - **WARNING!** Do not edit or delete the value Data Loader sets for the **anon-salt** options. If you try to modify a salt value, GPR generates an alarm message and Data Loader restores the original salt value. *If for some reason, Data Loader cannot restore the original salt value, your predictors become unusable for scoring and routing.* To rectify this situation you must recreate the agent and customer profiles, reload all interaction datasets, and retrain your models. If you do not recreate the agent and customer profiles and datasets exactly, you must also create and train new predictors and models. Therefore, Genesys strongly recommends that you do not modify or delete the salt values.
2. Before uploading the dataset to the GPR Core Platform, Data Loader uses this salt to anonymize the fields you specified as sensitive or PII data when you configured the schema.
3. The anonymized data is uploaded to the GPR Core Platform using TLS for secure data transport. The uploaded data is used for creating predictors and models.
4. After you create a predictor and one or more models, and begin using them to route interactions, the GPR subroutines retrieve the list of sensitive or PII features that are included in the active predictor. This list of features is stored in the URS Global Map.
5. The GPR Subroutines access the on-premises instance of your data to use in scoring requests. As a result, the Subroutines anonymize all sensitive fields included in the predictor you are using for scoring, based on the salt value stored in the Predictive_Route_DataCfg `Transaction List` object.
6. If one of the anonymized fields is the EMPLOYEE_ID, after the ActivatePredictiveRouting subroutine receives the response to the score request, it maps the agent scores back to the non-anonymized versions of the employee IDs so that routing can proceed.

7. Before the GPRIxnCleanup subroutine reports the routing outcome to the GPR Core Platform, it anonymizes all fields marked as PII that are included in the score outcome report. It then sends the results to the score log, which is stored in the cloud.

Unsupported characters in column names

The following characters are not supported for column names. If GPR encounters these characters in a .csv file, it reads them as column delimiters and parses the data accordingly.

- The pipe character
- `\t` (the TAB character)
- `,` (the comma)

Workaround: To use these characters in column names, add double quotation marks (" ") around the entire affected column name, except in the following situations:

- If you have a comma-delimited .csv file, add double quotations marks around commas within column names; you do *not* need quotations for the \t (TAB) character.
- If you have a TAB-delimited .csv file, add double quotations marks around TAB characters within column names; you do *not* need quotations for the , (comma) character.
- You must *always* use double quotations for the pipe character.

Data retention policies

GPR follows standard Genesys data retention guidelines for Genesys Multicloud CX as outlined in Section 14 of the Genesys Multicloud CX User Guide.

Most objects and data are deleted automatically after 90 days during which they are inactive. These include the following:

- Dataset data and the dataset object - Deleted after 90 days of idle time, which means no new files were appended and the dataset was not used to generate any data for predictors in that period.
- File upload object - Deleted after 90 days of idle time. Here idle time means this file was not used to generate any data for predictors in that period.
- Agent / Customer Profiles - Deleted after 90 days of idle time, which means the profile was not updated in the last 90 days.
- Model - Deleted after 90 days of idle time, which means the model was not used for any score requests in last 90 days.
- Predictor generated data and the predictor object - Deleted after 90 days of idle time, which means that no associated model was used for a score request in last 90 days.

The following data uses different retention policies:

- Uploaded anonymized files - Deleted 7 days after upload.

- Files stored for billing purposes - Deleted 60 days after creation.

Predictive Routing support for GDPR

Contents

This page describes product-specific aspects of Predictive Routing support for the European Union's General Data Protection Regulation (GDPR) in hybrid and Genesys Multicloud CX deployments.

Related documentation:

-

Data anonymization

PII, or personally identifiable information, and sensitive data, such as passwords, must be hidden when you upload it to the GPR Core platform. To ensure that sensitive data is secured, instruct Data Loader to anonymize the fields containing such data. For more information, see Data anonymization.

Data at rest

Data at rest is data which is not actively moving within the system or network and does not interact with any third-party applications such as data stored in hard drives, mobile phones, flash drives, laptop, and so on.

The data GPR uses for predictor or model creation is anonymized before it is uploaded. As a result, GPR does not store, access, or use non-anonymized data.

Data in transit

Data in transit, or data in motion, is data actively moving from one location to another such as across the internet or through a private network. GPR supports TLS v1.2 for the data in transit.

For more information, see Secure connections.

Data retention policies

GPR follows standard Genesys data retention guidelines for Genesys Multicloud CX as outlined in Section 14 of the Genesys Multicloud CX User Guide.

Most objects and data are deleted automatically after 90 days during which they are inactive. For more information, see Data retention policies.

Updating or removing GDPR-related data

If you need to update, view, or delete any data that might be affected by GDPR controls, contact Genesys Professional Services for assistance.

Deploy the URS Strategy Subroutines

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.

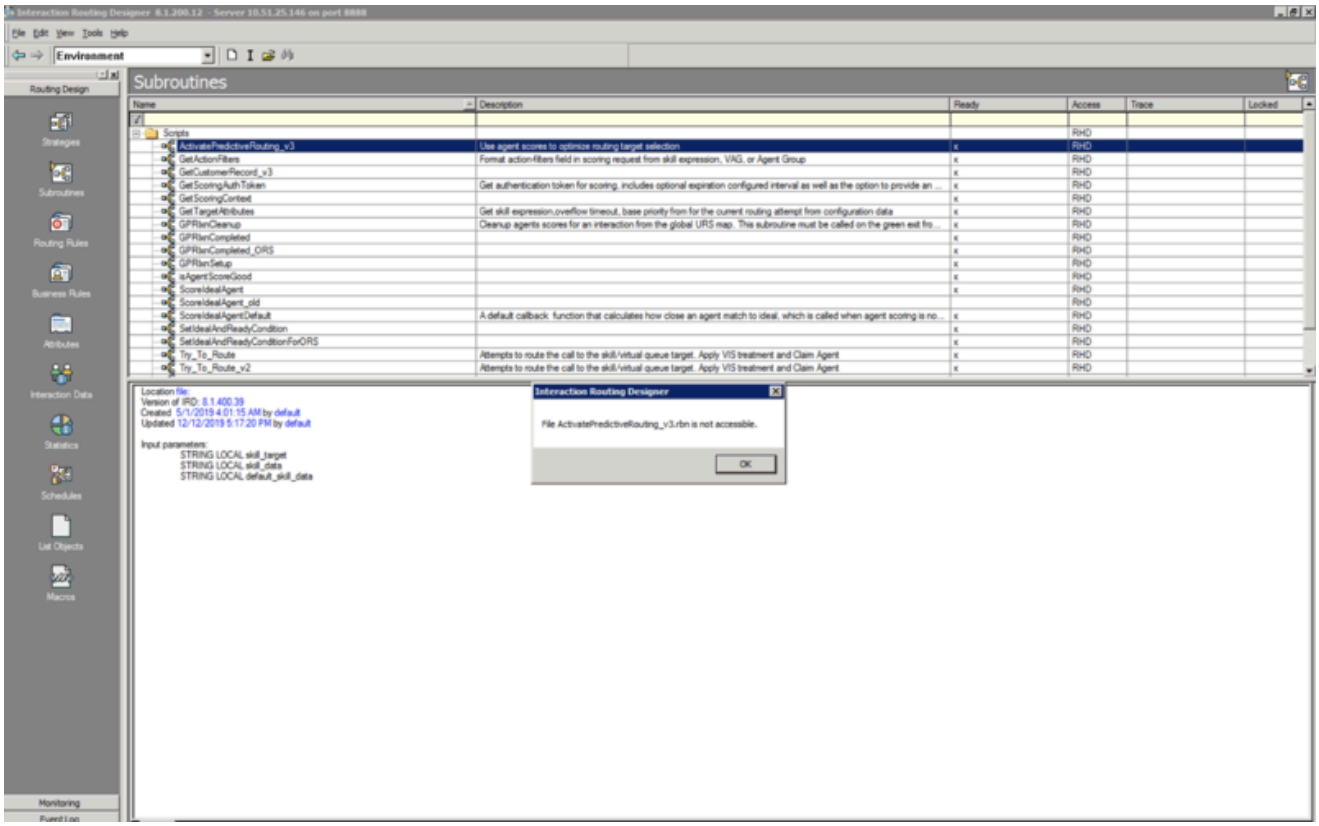
Install the subroutines

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.

Integrate with Genesys Reporting

Contents

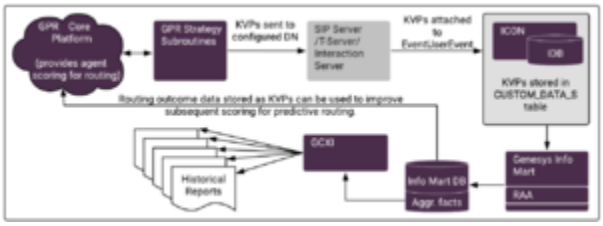
Feature coming soon

Genesys Predictive Routing (GPR) can supply a variety of information about routing outcomes for use by the Genesys reporting applications. GPR sends data for historical reporting in Key-Value Pair. This KVP data, which is stored in the Info Mart database, can also to be fed back into GPR to refine predictors. In addition, Stat Server sends this KVP data to Pulse for real-time reporting.

Related documentation:

-

GPR Reporting Data Flow



GPR Reporting Data Flow

1. GPR data in the form of KVPs is attached to EventUserEvent TEvents.
   The UserEvent contains **AttributeThisDN** with a value of `Route Point`, which identifies where the strategy is executed and **AttributeUserData,** which holds a list of KVPs containing data about the interaction.

   This Route Point should also be specified in the ***vq-for-reporting*** option.
2. Interaction Concentrator stores the KVP data in the Interaction Database (IDB), in the CUSTOM_DATA_S table.
3. Genesys Info Mart gathers this raw data from IDB and prepares it for use in Genesys CX Insights historical reporting on GPR activity and performance.
   - Genesys Info Mart has specific requirements for the KVPs that must be attached (see GPR KVPs for Genesys Reporting, below).
4. Reporting and Analytics Aggregates (RAA) further transforms the data in preparation for use by the presentation layer.
5. Using Genesys CX Insights, you can create the following reports based on the GPR data. These reports are described in the *Genesys Customer Experience Insights User's Guide*:
   - Predictive Routing - AHT & Queue Dashboard
   - Predictive Routing - Model Efficiency Dashboard
   - Predictive Routing Agent Occupancy Dashboard

- Predictive Routing A/B Testing Report
- Predictive Routing Detail Report
- Predictive Routing Operational Report
- Predictive Routing Queue Statistics Report

6. Using **Pulse**, you can access the *Agent Group KPIs by Predictive Model* and *Queue KPIs by Predictive Model* templates for real-time reporting. These templates are available from the Genesys Dashboard Community Center.
   NOTE: Because Place objects do not have a Logged Out status, unlike Agent objects, some discrepancies appear in the Logged In, % Ready, and % Not Ready statistics in the GPR Agent KPI template between reports run on the Agent Group object and the Place Group object. By default, the Place object takes the NotReadyForNextCall status. As a result, a Pulse GPR Agent KPI report on a Place Group object does the following:

   - Displays all places in the Place Group as Logged In and in Not Ready, even though the associated DN is not logged in.
   - If an agent is logged out from the associated DN, the status of the Place is changed to NotReadyForNextCall from WaitForNextCall.

In addition to the powerful Genesys Reporting solution, the GPR web application offers various reports and windows enabling you to Monitor trends and performance (described in the *Predictive Routing Help*).

The following topics provide in-depth information about how Universal Routing Server (URS) makes routing decisions when you are using Predictive Routing and how GPR scores agents in various scenarios. Use the material covered in these topics to inform your understanding about what data the KVPs described in this topic store and how to create the most useful reports from the available data:

- Routing Scenarios Using GPR
- How Does GPR Score Agents?

Configure Historical Reporting

Genesys Info Mart release 8.5.014.19 or higher and later provides support for GPR reporting out-of-box in a Hybrid environment, with no additional configuration required on the Genesys Info Mart side. However, to send GPR data to Genesys Info Mart, as well as to see GPR data in Genesys CX Insights reports, you need to modify the configuration of GPR, Interaction Concentrator, and Reporting & Analytics Aggregates (RAA, the aggregation engine hosted by Genesys Info Mart).

> ### Tip
> For general information about how Genesys Info Mart uses attached user data, see the Genesys Info Mart and Attached User Data section of the *Genesys Info Mart*

> *Deployment Guide.*

1. Ensure that ICON and IDB have been deployed as Genesys Info Mart requires, and that ICON is connected to the T-Server(s) or SIP Server(s) handling the GPR interactions. For details, see Preparing Interaction Concentrator in the *Genesys Info Mart Deployment Guide.*
2. Configure a DN for GPR reporting data.
   - Open Genesys Administrator Extension (GAX) and specify a DN to use with GPR. This DN can be a VQ DN, a Trunk Group DN, or any other recognized type, as long as you configure ICON to monitor it. The name of the DN is used inside URS strategy subroutines, so it should be meaningful and recognizable.
3. Configure URS:
   - In the connections of your Universal Routing Server application, add the T-Server/SIP Server used to define the reporting Switch and DN in the GPR configuration. For example, GPR_Switch.
4. In IRD, set up your routing solution to attach the required KVPs in UserEvents. For an example to guide you, refer to the GPRIxnCompleted subroutine provided with GPR.
5. Configure Interaction Concentrator to store the GPR KVPs:
   1. Set the store-event-data option to `all`, the recommended setting in GPR deployments, or `conf`. This option controls which KVP data from AttributeUserData of EventUserEvent ICON stores in the G_CUSTOM_DATA_S table.
      - To simplify configuration in deployments where GPR data is extracted for reporting, Genesys recommends setting the **[custom-states].store-event-data** configuration option to `all`, which ensures that ICON stores all the UserEvent-based KVPs that Genesys Info Mart requires. However, be aware that setting **store-event-data**=`all` has performance and security implications:
        - Performance — Processing and storing a large number of UserEvent-based KVPs increases database resource requirements and can impact performance.
        - Security — Sensitive data (for example, credit card information) might be sent in UserEvents that are not used for reporting. Unlike the situation for call-based attached data, where the G_SECURE_USERDATA_HISTORY table is available to provide secure IDB storage, there is no secure IDB table parallel to G_CUSTOM_DATA_S that provides separate, secure storage for sensitive data.
      - If you set **store-event-data** to `conf`, use the following EventData options to specify which KVPs to store:
        - EventData=char,gpmVQDBID,char,gpmGlobalScore,char,gpmMedianScore, char,gpmAdjustedAgentScore,char,gpmVQGUID
        - EventData_1=char,gpmModelId,char,gpmMinScore,char,gpmDefaultScoreUsed, char,gpmTargetSize,char,gpmWaitTime
        - EventData_2=char,gpmAgentRank,char,gpmUse,char,gpmVQNumber, char,gpmRouteAttemptId,char,gpmPredictorType
        - EventData_3=char,gpmPredictorId,char,gpmScoreAboveMedian, char,gpmSuitableAgentsCount,char,gpmDefaultScoredAgents,char,gpmPredictor
        - EventData_4=char,gpmInitialScoreThreshold,char,gpmMessage, char,gpmResult,char,gpmMode,char,gpmCustomerFound

- EventData_5=char,gpmSwitchName,char,gpmMaxScore, char,gpmDefaultAgentScore,char,gpmAgentScore,char,gpmAgentID
- EventData_6=char,gpmAgentDBID,char,gpmModel,char,gpmGlobalScoreCount, char,gpmPriorityIncrement,char,gpmFinalScoreThreshold
- EventData_7=char,CALLID,char,START_TS,char,ServiceType, char,CustomerID,char,context_id
- EventData_8=char,gpmStatus,char,gpmRoutingMethod

2. Ensure that you have added the T-Server/SIP Server corresponding to the DN you created earlier for GPR to the **Connections** tab of the Interaction Concentrator `Application` object.

6. Configure GPR to attach KVP data by configuring the following options on the **Predictive_Route_DataCfg** Transaction List object:
   - send-user-event - Enables attaching the Predictive Routing-specific key-value pairs.
   - vq-for-reporting - Indicates the virtual queue or DN where URS sends the GPR user event data. The user event data, in the form of key-value pairs (KVPs), is attached to EventUserEvent in the AttributeUserData attribute. For the list of KVPs to be attached, see GPR KVPs for Genesys Reporting, below. The following KVPs are mandatory for data to be available for Genesys Reporting:
     - gpmResult
     - CALLID
     - START_TS
     - ADDED_TS
7. Ensure that your deployment has been configured as required for Genesys Info Mart to support reporting on contact center activity in general. For a summary of the configuration requirements, see Enabling Reporting on Voice Activity in the *Genesys Info Mart Deployment Guide*.
8. Enable aggregation of GPR data. (Required for Genesys CX Insights reporting or other applications that use RAA aggregation.)
   - In the **[agg-feature]** section on the Genesys Info Mart application object, specify the enable-gpr and enable-gpr-fcr options.
9. Verify the reporting data chain. After a few interactions have been routed with GPR in an operational mode, verify that the required KVPs are being sent, stored, and used as expected:
   - Check the T-Server/SIP Server logs to verify that UserEvents are being sent with the required KVPs.
   - Check the ICON logs and the G_CUSTOM_DATA_S table in IDB to verify that ICON is recording the required KVPs.
   - Check the GPM_* tables in the Info Mart database to verify that Genesys Info Mart is correctly transforming the data.

For more information about configuring user data storage in Interaction Concentrator to work with Genesys Info Mart, see Important custom-states ICON Configuration Options and Configuration Considerations in the *Genesys Info Mart Deployment Guide.*

GPR KVPs for Genesys Reporting

The following table describes the KVPs that Genesys Info Mart uses to enable

GPR reporting.

> **Tip**
>
> - Use the Search box to quickly locate a specific KVP.
> - Although the Predictive Routing short name is GPR, the gpm* prefix shown in the table below is correct. It reflects an earlier name for the product.

KVP	Description	KVP Type	Info Mart Database Target
KVP	**Description**	**KVP Type**	**Info Mart Database Target**

"> ADDED_TS
UTC timestamp, indicating the date and time when the record was added as inherited from the T-Server TEvent.
Default value: no default value
Valid values: any valid UTC timestamp

Note: This KVP is mandatory for Genesys Info Mart reporting. INT GPM_FACT.ADDED_TS"> CALLID
Value of AttributeCallUUID for the interaction.
Default value: a valid CALLID

Note: This KVP is mandatory for Genesys Info Mart reporting. CHAR(32) GPM_FACT.MEDIA_SERVER_IXN_GUID"> CustomerID

Introduced: 9.0.016.00 The GPRIxnCleanup subroutine takes this KVP from user data attached to the interaction, and passes it to the Genesys Historical Reporting solution in the EventUserEvent event. GPR does not generate this KVP. Postgres: varchar(255); Oracle: VARCHAR2(255 CHAR); Microsoft SQL: varchar(255)/nvarchar(255) IRF_USER_DATA_GEN_1.CUSTOMER_ID"> gpmAdjustedAgentScore

Introduced: 9.0.015.00 The final agent score used to route the associated interaction to the selected agent. This score is calculated from the gpmAgentScore combined with any agent occupancy factor.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.ADJUSTED_SCORE"> gpmAgentDBID
Optional. The DBID of the agent to whom the interaction was routed.
Default value: no default value INT RESOURCE_.RESOURCE_CFG_DBID

(referenced through GPM_FACT.RESOURCE_KEY)"> gpmAgentRank
The rank of the agents in the target group, based on agent scores sorted in descending order.
Default value: 0
Valid values: 0, any positive integer SHORT GPM_FACT.AGENT_RANK"> gpmAgentScore
The score of the agent to whom the interaction was routed.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.AGENT_SCORE"> gpmCustomerFound
Indicates whether features from the customer record specified in the routing strategy were successfully retrieved from the Customer Profile schema uploaded to the AI Core Services and used to calculate agent scores.
Default value: unknown
Valid values: 0 (= No), 1 (= Yes), unknown Enum GPM_RESULT.CUSTOMER_FOUND (referenced through GPM_FACT.GPM_RESULT_KEY)"> gpmDefaultAgentScore

Introduced: 9.0.015.00 This default agent score for the associated interaction. The value is the outcome, for this interaction, of the setting specified in the default-agent-score configuration option.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.DEFAULT_SCORE"> gpmDefaultScoredAgents

Introduced: 9.0.015.00 The number of agents assigned the default score for the associated interaction.
Default value: 0
Valid values: 0, any positive integer INT GPM_FACT.DEFAULT_SCORES_COUNT"> gpmDefaultScoreUsed

Introduced: 9.0.015.00

- 0 - The agent score for the associated interaction is taken from the scoring response returned by GPR.
- 1 - The agent score for the associated interaction is calculated based on the value set for the default-agent-score configuration option.

Default value: 0
Valid values: 0, 1 INT GPM_FACT.DEFAULT_SCORE_USED"> gpmDeploymentType

Introduced: 9.0.017.00 The URS Strategy Subroutines read the value for this KVP from the **deployment-type** configuration option, which is set once, when a new Predictive Routing account is initially configured.

Note: This KVP is not currently stored as a separate column in the Genesys Info Mart database. It can be accessed from the score_log file using the GPR API.

Default value: hybrid
Valid values: hybrid, cloud N/A N/A"> gpmFinalScoreThreshold

Introduced: 9.0.015.00 The final threshold value used to route the associated interaction to the selected agent. The routing strategy calculates the value from the configured score threshold combined with values resulting from any agent holdout options.
Default value: 0
Valid values: any integer INT GPM_FACT.FINAL_SCORE_THRESHOLD">
gpmGlobalScore
The mean score calculated for an interaction using the Global Model.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.GLOBAL_SCORE">
gpmGlobalScoreCount

Introduced: 9.0.015.00 Describes the number of agent scores returned for an interaction using a GLOBAL model.
Default value: 0
Valid values: 0, any positive integer INT GPM_FACT.GLOBAL_SCORES_COUNT">
gpmInitialScoreThreshold

Introduced: 9.0.015.00 The initial threshold value used for the interaction, taken from the value set in the score-base-threshold configuration option.
Default value: 0
Valid values: any integer INT GPM_FACT.INITIAL_SCORE_THRESHOLD">
gpmMaxScore
The score of the best-matching agent in the target group.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.MAX_SCORE">
gpmMedianScore
The median score for the target group of agents to which the agent who received the interaction belongs.
Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.MEDIAN_SCORE">
gpmMessage
The message that displays when the Predictive Routing result reported in the gpmResult KVP is an error.
Default value: no default value CHAR(255) GPM_FACT.MESSAGE">
gpmMinScore
The score of the worst-matching agent in the target group.

Default value: 0
Valid values: any non-negative float value FLOAT GPM_FACT.MIN_SCORE"> gpmMode

Modified: 9.0.015.00 - The value `off` was added. The mode in which Predictive Routing is operating, as specified by the prr-mode configuration option. For information about turning predictive routing off, see Turn Off Predictive Routing.
Default value: unknown
Valid values: prod, off, dry-run, ab-test-time-sliced, unknown Enum GPM_RESULT.GPM_MODE (referenced through GPM_FACT.GPM_RESULT_KEY)"> gpmModel
The name of the Model used to calculate agent scores for the interaction.
Default value: unknown
Valid values: The name of any Model in your environment CHAR(255) GPM_MODEL.MODEL (referenced through GPM_FACT.GPM_MODEL_KEY)"> gpmModelId
The UUID of the Model used to calculate agent scores for the interaction.
Default value: unknown
Valid values: The ID for any Model in your environment CHAR(24) GPM_MODEL.MODEL_ID (referenced through GPM_FACT.GPM_MODEL_KEY)"> gpmPredictor
The name of the Predictor. If an error is encountered, the section name specified in the **Predictive_Route_DataCfg** Transaction List object is used as the Predictor name.
Default value: unknown
Valid values: The name of any Predictor in your environment CHAR(255) GPM_PREDICTOR.PREDICTOR (referenced through GPM_FACT.GPM_PREDICTOR_KEY)"> gpmPredictorId
The UUID of the Predictor used for scoring.
Default value: unknown
Valid values: The ID for any Predictor in your environment CHAR(24) GPM_PREDICTOR.PREDICTOR_ID (referenced through GPM_FACT.GPM_PREDICTOR_KEY)"> gpmPredictorType

Introduced: 9.0.016.00 Describes whether the applied predictor is used for Sales KPI or Service KPI.
Default value: unknown
Valid values: Sales, Service CHAR[32] GPM_DIM1.PREDICTOR_TYPE"> gpmPriorityIncrement

Introduced: 9.0.016.00 If the value is 0, the priority of the interaction did not increase above the configured base_priority value. If the value is 1, the priority of the interaction did increase above the configured base_priority and, as a result,

the selected agent was not verified for the expected threshold score.

Note: This KVP is not currently stored as a separate column in the Genesys Info Mart database. It can be accessed from the score_log file using the GPR API.

Default value: 0
Valid values: 0,1 N/A N/A"> gpmPriorityIncrement

Introduced: 9.0.016.00 Specifies whether the priority of the interaction was increased above the configured base priority value. If set to 0, the interaction was routed at the base priority. If the value for this KVP is 1, the priority was incremented above the base priority and, as a result, the selected agent was not verified for the expected threshold score.

- This KVP is sent only to the score_log file. It is not stored in any Genesys Info Mart table.

Default value: 0
Valid values: 0, 1 CHAR[32] Not mapped"> gpmResult

Modified: 9.0.015.00 - The values 12, 13, 14, and 15 were added. 9.0.018.00 - The value 16 was added. 9.0.018.01 - The value 17 was added. The result of Predictive Routing processing. If there is an error, the gpmMessage KVP contains the error message.

- 1 - Ok
- 2 - Authentication to scoring engine failed
- 3 - Scoring request failed
- 4 - Agent list is empty
- 5 - URS overload, interaction skipped
- 6 - Predictor not found
- 7 - Failed to build scoring request
- 8 - SetIdealAgent or SetReadyCondition execution error
- 9 - Interaction log not found in global map
- 10 - Unknown error
- 11 - Channel is not supported
- 12 - Reserved for future use
- 13 - Call Abandoned
- 14 - Call Routing Failed
- 15 - Predictive Routing is turned off or not used for this interaction

- 16 - No agents found with a score above minimum threshold
- 17 - Empty WFM target list

Default value: no default value
Valid values: 1–17

Note: This KVP is mandatory for Genesys Info Mart reporting. Enum GPM_RESULT.GPM_RESULT (referenced through GPM_FACT.GPM_RESULT_KEY)"> gpmRouteAttemptId
The sequence number of the attempt to route an interaction using Predictive Routing. The value of this KVP is incremented each time the ActivatePredictiveRouting subroutine is called by the strategy, starting from 1.
Default value: 0
Valid values: integers starting from 1 INT GPM_FACT.ROUTE_ATTEMPT_ID"> gpmRoutingMethod

Introduced: 9.0.015.00 Reserved for future use.
Default value: unknown CHAR[32] GPM_DIM1.ROUTING_CRITERIA"> gpmScoreAboveMedian
Indicates whether the score for the selected agent was better than the median score for the target group.
Default value: unknown
Valid values: 0 (no), 1 (yes), unknown Enum GPM_FACT.SCORE_ABOVE_MEDIAN"> gpmStatus
Indicates the scenario under which the interaction was processed. For more information about the scenarios, see Routing Scenarios Using GPR.
Default value: unknown
Valid values: agent-surplus, call-surplus, unknown Enum GPM_RESULT.GPM_STATUS (referenced through GPM_FACT.GPM_RESULT_KEY)"> gpmSuitableAgentsCount

Introduced: 9.0.015.00 The number of agents who had scores greater than or equal to the initial threshold value when the scoring response was received.
Default value: 0
Valid values: 0, any positive integer INT GPM_FACT.SUITABLE_AGENTS_COUNT"> gpmTargetSize
The size of the scored target group (in other words, the length of the list of agents received from the scoring engine).
Default value: 0
Valid values: 0, any positive integer SHORT GPM_FACT.TARGET_SIZE"> gpmUse
The meaning depends on the mode in which Predictive Routing is operating (see the description of the gpmMode KVP). This field is set to one of the following

values:

- `1` - When the mode is `ab-test-time-sliced`, indicates that the interaction was selected for Predictive Routing. When the mode is `prod`, indicates the normal case, when Predictive Routing occurred without error.
- `0` - When the mode is `ab-test-time-sliced`, indicates the interaction was processed with skill-based routing. When the mode is `dry-run`, indicates that the interaction completed without error.
- `unknown` - For any mode, indicates that an error occurred in one of the Predictive Routing subroutines, and the solution defaulted to skill-based routing.

Default value: unknown
Valid values: 1, 0, unknown Enum GPM_RESULT.GPM_USE (referenced through GPM_FACT.GPM_RESULT_KEY)"> gpmVQDBID

Introduced: 9.0.016.00 The DBID of the virtual queue or DN configured in the vq-for-reporting configuration option (configured on the Predictive_Route_DataCfgTransaction List object).

- Requires Genesys Info Mart release 8.5.014.19 or higher.

Default value: No default value
Valid values: Any valid DBID INT RESOURCE_.RESOURCE_CFG_DBID (referenced through GPM_FACT.VQ_RESOURCE_KEY)"> gpmVQGUID

Introduced: 9.0.016.00 Value of the Virtual Queue ID (RPVQID) stored in the interaction user data. This is a special GUID value that uniquely identifies the entrance of the interaction into certain virtual queues. The RPVQID is created by URS when the interaction enters into the virtual queue and is present in all VirtualQueue events that URS distributes.

- Requires Genesys Info Mart release 8.5.014.19 or higher.

Default value: No default value
Valid values: Any valid Virtual Queue GUID CHAR[32] GPM_FACT.VQ_GUID"> gpmWaitTime
The amount of time, in seconds, the interaction spent in the queue used for Predictive Routing decision-making, starting from when the strategy started to process the interaction until it was routed to the agent. Note that the point when processing starts might depend on how you have configured your strategy.
Default value: 0
Valid values: 0, any positive integer INT GPM_FACT.WAIT_TIME"> ServiceType

Introduced: 9.0.016.00 The GPRIxnCleanup subroutine takes this KVP from user data attached to the interaction, and passes it to the Genesys Historical Reporting solution in the EventUserEvent event. GPR does not generate this KVP. Oracle: VARCHAR2(255 CHAR); Postgres: varchar(255); Microsoft SQL: nvarchar(170) INTERACTION_DESCRIPTOR.SERVICE_TYPE"> START_TS UTC timestamp, indicating the time when the interaction arrived at the contact center.

Note that this value is different from gpm-ixn-timestamp, which, in release 9.0.014.04 and earlier, indicates the time when the strategy started processing the interaction. gpm-ixn-timestamp is configured in the default_skill_data object, from which it is passed to the ActivatePredictiveRouting_v3 subroutine.

In URS Strategy Subroutines 9.0.015.00 and higher, gpm-ixn-timestamp is not used, and START_TS must be passed in the default_skill_data parameter. gpmWaitTime (the actual wait time of the interaction in the queue before an agent is selected) is calculated based on the difference between the UTC time when agent is selected minus the START_TS value.

Default value: no default value
Valid values: a valid UTC timestamp

Note: This KVP is mandatory for Genesys Info Mart reporting. INT GPM_FACT.START_DATE_TIME_KEY

How GPR reports data for billing

Contents

GPR generates data that is used to determine which interactions are billable.

Related documentation:

-

What is billed?

GPR Billing counts only successfully routed interactions. If an interaction is not delivered to an agent, it is not be billed, even if it was routed using GPR. The following points describe how GPR handles some specific scenarios:

- If your routing strategy is configured to ask GPR to score agents multiple time for a single interaction, you are billed for each score request that was successfully processed, routed, and answered. For example, your strategy might transfer an interaction using GPR, resulting in scoring for the initial agent and a second scoring request to locate the agent to which the interaction is transferred. Or your routing might use queue chains, with multiple calls to GPR for agent scores.
- If a consult call has the same ConnectionID as the main one, the consult is to be billed as only one call. If the consult call results in two ConnectionIDs, it is billed as two calls.
- An interaction scored in queue by GPR that ends up being routed using skills-based routing—for example, because it remained in queue long enough to trigger default routing—the interaction **is** charged because GPR is statistically contributing to the lift of the whole queue by the fact that *all* interactions are scored. Details on this scenario (and all others) is captured in the gpmMode, gpmResult, and gpmUse KVPs.

Start and stop GPR on-premises components

Contents

This topic explains how to start and stop Data Loader, and how to enable or disable use of Predictive Routing when routing interactions.

Related documentation:

-

Start and stop Data Loader

Data Loader is deployed in a container that also contains Local Control Agent, which communicates with Genesys Administrator to start, monitor status of, and stop Data Loader.

All Data Loader instances can be started and stopped from Genesys Administrator. For instructions on starting and stopping application from Genesys Administrator, see the System Dashboard topic in the *Genesys Administrator Help*.

To start the container where Data Loader is located, run the following scripts from the installation directory to start the container:

```
$ cd ai-data-loader-scripts/scripts/
```

```
$ bash start.sh
```

Only Local Control Agent (LCA) is started when container is started. Once the Data Loader host is shown as up and running in Genesys Administrator, you can start Data Loader from there as you would any other Genesys component.

- See How to Start and Stop Applications and Solutions for details about LCA and monitoring solutions in Genesys Administrator.

To stop Data Loader, execute the **stop.sh** command from the **/scripts** folder.

To restart Data Loader container, execute the **restart.sh** script from the **/scripts** folder.

Start and stop use of the URS Strategy Subroutines

To turn on Predictive Routing in your routing strategy:

1. Open the Predictive_Route_CfgData Transaction List object.
2. Set the ***prr-mode*** option to any value *except* `off` in all sections that define Predictors, and also in the **[default-predictor]** section.

To turn off Predictive Routing in your routing strategy:

1. Open the Predictive_Route_CfgData Transaction List object.
2. Set the ***prr-mode*** option to `off` in all sections that define Predictors, and also in the **[default-predictor]** section.

Routing scenarios using GPR

Contents

- Administrator

Learn about Predictive Routing interaction flows, how the URS Strategy Subroutines work together to score agents and identify a routing target, how URS ranks agents by score, and how GPR handles agent reservation.

Related documentation:

-

If you would like to evaluate Genesys Predictive Routing for use with service-level routing or business-objective routing, contact Genesys Professional Services for a review of your routing environment.

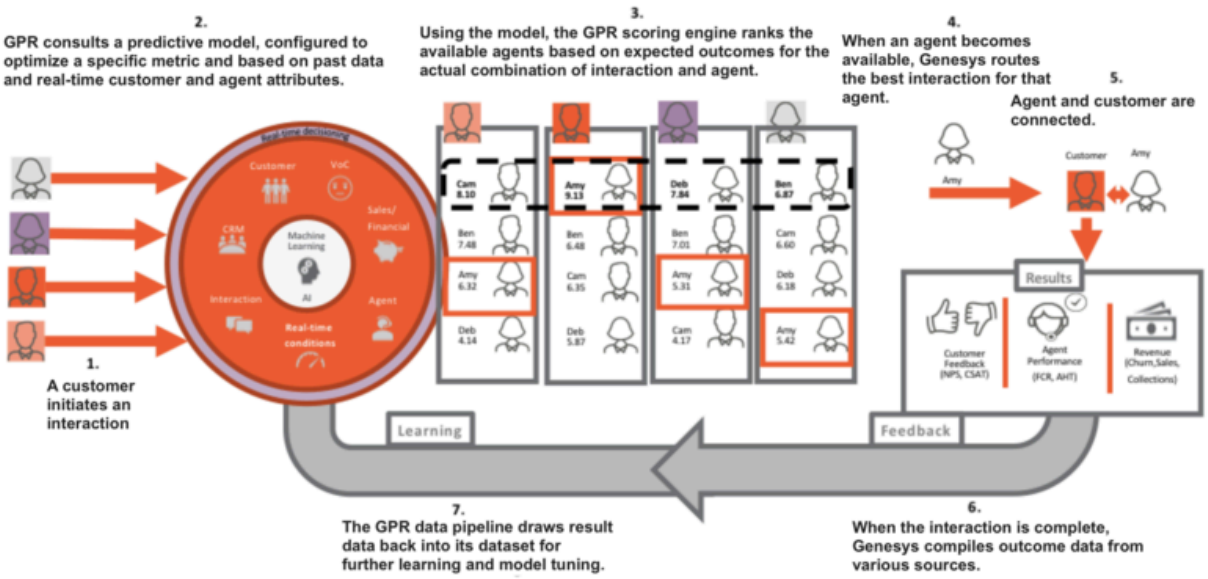
If your environment uses multiple URS instances receiving interactions from a single T-Server, the only criterion used to select the next interaction for routing is priority.

This topic assumes that you are using a virtual agent group (VAG) as the target for your routing. If you route using a skill expression to identify your targets, convert it to a VAG string expression using the IRD MultiSkill or CreateSkillGroup function before passing the resulting string as an argument to the ActivatePredictiveRouting subroutine. See Using Agent Skills for Ideal Agent Selection in the *Supplement to the Universal Routing 8.1 Reference Manual* for more information.

High-Level Predictive Routing interaction flow

The graphic in this section shows a very general interaction flow using Predictive Routing. Refinements to the flow depend greatly on details of your environment. Key aspects that differ in various environments:

- Your data - That is, the interaction types supported and the applications that might have relevant information. Genesys Info Mart is a key data source, but CRM systems and other applications in your environment can also provide important data. See Set up data for import for more information.
- Your pre-routing data flow - This depends on the interaction type and the exact architecture in your environment. For example, is this a chat interaction or a call? Do you use an IVR, and if so, what information do you attach?
- The Genesys routing solution you are using - Predictive Routing supports routing with IRD/URS.
- Your reporting solution for Predictive Routing - Whether you are using GCXI, Genesys Pulse, or another solution to present the data stored in Genesys Info Mart.



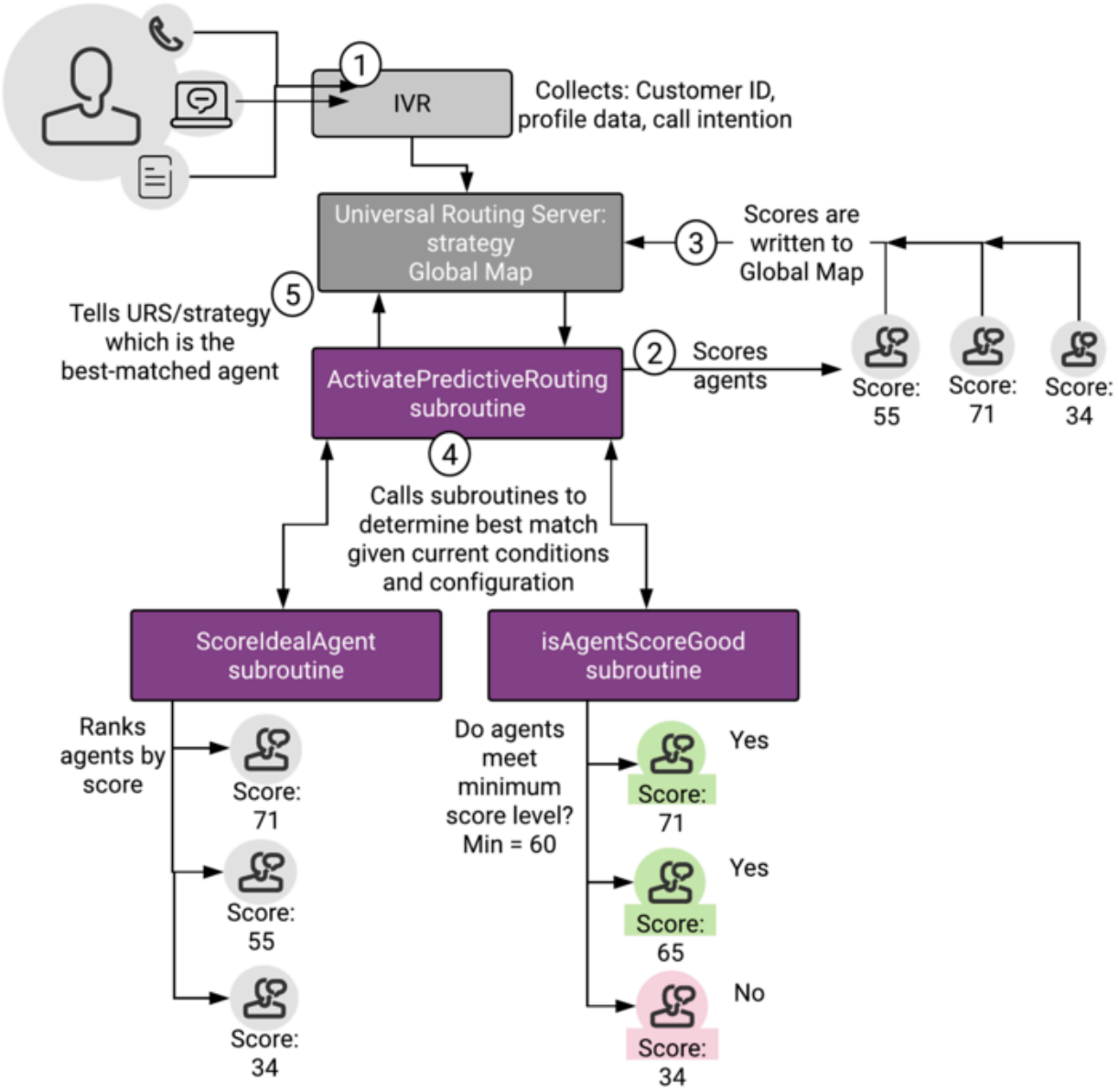
How the Strategy Subroutines work

The following sequence provides a basic overview of the way the various GPR subroutines work together to evaluate agent scores and determine the best match given the currently available agents and the currently waiting interactions.

1. ActivatePredictiveRouting retrieves agent scores from the GPR Core Platform via REST API request and stores them in the global map in URS memory. The name of the map is the interaction ConnectionId (the original ID, if the interaction is a consult). This map contains pairs of agent employee IDs as the keys and their scores for the interaction as values. ActivatePredictiveRouting calls the SetIdealAndReadyCondition subroutine for further interaction processing.
2. SetIdealAndReadyCondition processes the different modes of Predictive Routing. It calls the SetIdealAgent IRD function to schedule the execution of the URS callback subroutines. It calls the ScoreIdealAgent subroutine to facilitate interaction queueing according to their scores, and calls SetReadyCondition (if enabled) to call the isAgentScoreGood subroutine.
   The parameters for these callback subroutines are retrieved and verified before any URS request is invoked to enable the callbacks.
3. After establishing a list of potential targets based on the target expression (Skill, Agent Group, and so on) SetIdealAndReadyCondition then executes the ScoreIdealAgent callback subroutine.
4. ScoreIdealAgent retrieves the scores for the potential target agents from the global map set in Step 1.
5. When an agent becomes ready, URS executes the isAgentScoreGood subroutine to determine whether that target is acceptable. If you enabled agent hold-out, URS executes the isAgentScoreGood subroutine when an agent becomes ready, which determines whether that agent reaches the specified threshold score. If not, URS waits for a configured timeout period, then checks whether any agent now satisfies the adjusted threshold value. See How Does GPR Score Agents? for a detailed discussion of how agent hold-out routing works.
6. Once the isAgentScoreGood subroutine locates an available agent who scores above the current threshold, it sends the target details to URS to initiate routing.

7. URS calls the GPRIxnCompleted subroutine as a custom step from a Routing Block in the strategy. It collects the Predictive Routing outcome for the successfully routed interaction (the DBID of the agent to whom it was distributed, the score of the agent, other interaction statistics relevant for the Predictive Routing performance) and prepares the user data for Predictive Routing reporting.
8. URS calls the GPRIxnCleanup subroutine from both the success and failure exits from the Routing block, or if the interaction is abandoned. The purpose of the subroutine is to publish the Predictive Routing reporting user data and to clean up the ScoreIdealAgent and isAgentScoreGood callback subroutines. GPRIxnCleanup publishes reporting data in two ways:
   - It sends a UserEvent containing the user data relevant for Predictive Routing to T-Server/SIP Server, from which is enters the Genesys historic reporting solution flow.
   - It can submit the same data AI Core Services via REST API request where it is stored in the score_log and can be retrieve using an API request.

> **Important**
>
> If your routing strategy uses the SelectDN and SuspendDN IRD functions instead of a Routing Block, consult with Genesys Professional services about how the ActivatePredictiveRouting, GPRIxnCompleted and GPRIxnCleanup subroutines can be integrated into your strategy.



Routing scenarios using Predictive Routing

When you are using Predictive Routing to route interactions, there are two main scenarios that affect how this matching plays out:

- **Agent Surplus** - There are relatively few interactions, which means there could be a number of high-score agents available. You can configure a minimum threshold so that, if the agents available are not very highly ranked, the strategy keeps the interaction in queue until a better-scoring agent becomes

available.

- **Interaction Surplus** - There are many interactions, so that most agents are busy and it might be more difficult to find an ideal agent for each interaction. In such a scenario, you can have agents matched to the interaction for which they have the highest probability of getting a positive result.

Agent Surplus Flow

In this case there are agents logged in and in the Ready state who can respond to interactions immediately. From a Virtual Agent Group that is defined by skill expression, URS first tries to route an interaction to an agent with the best score, using the following process to match agents and interactions:

1. An interaction arrives at the routing strategy, which has a target group of agents.
2. The ActivatePredictiveRouting subroutine sends a request to the Predictive Routing scoring server via HTTP request.
3. Predictive Routing returns scores for each agent in the target group based on the criteria you selected in the active model.
4. The ActivatePredictiveRouting subroutine updates a global cache in URS memory, which keeps agent scores for all interactions. When URS tries to route the current interaction to the agent group, it sorts the agents according to their scores, in descending order, and routes to the agents with the best score first.

When URS takes an interaction from the queue:

1. URS calls the ScoreIdealAgent subroutine, which reads the agent scores in the target group from global map and ranks the agents by score.
2. URS calls the IsAgentScoreGood subroutine, which selects the available agent with the highest score, assuming the agent has a score high enough to be selected for this interaction.
   In an agent-surplus scenario, it is typically not a problem to route to an agent with a good score. For scenarios where this is not the case, see **Interaction Surplus Flow**, below.
3. URS calls the GPRIxnCompleted subroutine, which updates user data with the scoring result for storage in Genesys Info Mart.
4. URS calls the PRRLog macro, which logs the result in the URS log file.

Interaction Surplus Flow

This scenario covers situations when all agents are already busy handling interactions and new interactions are queued. When one of the agents becomes ready, the system selects the interaction for which the agent has the best score. This is not necessarily the interaction that has been in the queue longest.

When interactions are waiting, URS uses a number of criteria to decide the order in which it directs the interactions to the best target. In general, URS uses the following hierarchy:

1. Interaction priority.
2. Best agent score.

> **Important**
>
> In scenarios where both scored and unscored interactions might have the same priority, scoring is disregarded for all the interactions and the selection is based on the next differentiating criterion, time.

3. Time in queue, which can be based on age of interaction or time in queue and can incorporate predicted wait time.
4. Interaction ID (URS selects the interaction with the lowest—oldest—ID). This is a rarely-used "tie-breaker" criterion.

Using gpmStatus and gpmSuitableAgentsCount to Monitor Your Routing

gpmStatus and gpmSuitableAgentsCount are KVP values written in the Genesys Info Mart database when an interaction is routed using the GPR subroutines. (You can also retrieve the values by using the GPR API to query the score log.)

- gpmStatus indicates whether there was an agent-surplus or an interaction-surplus condition when the interaction was routed.
- gpmSuitableAgentsCount indicates the number of agents who have scores returned from AICS greater than or equal to the initial threshold value when the scoring response is received. If gpmSuitableAgentsCount is `0`, then no agents have eligible scores compared with the threshold value, so the interaction must wait for a higher-scoring agent to become available or for the next threshold relaxation step

These KVP values, when analyzed for different interactions over a representative day or week period can help you understand your contact center traffic and GPR performance. The following table indicates certain scenarios and how to interpret them.

KVP Values	Inference
gpmStatus = caller-surplus gpmSuitableAgentsCount > 0	Your GPR Model is returning useful scores with relation to the configured routing threshold, but agent staffing is not adequate to produce satisfactory wait times.
gpmStatus = agent-surplus gpmSuitableAgentsCount > 0 or consistently a very small number	Analyze why the scores GPR returns are not meeting the configured threshold. You might need to retrain your Model, adjust the scoring expression, or reduce the threshold level.

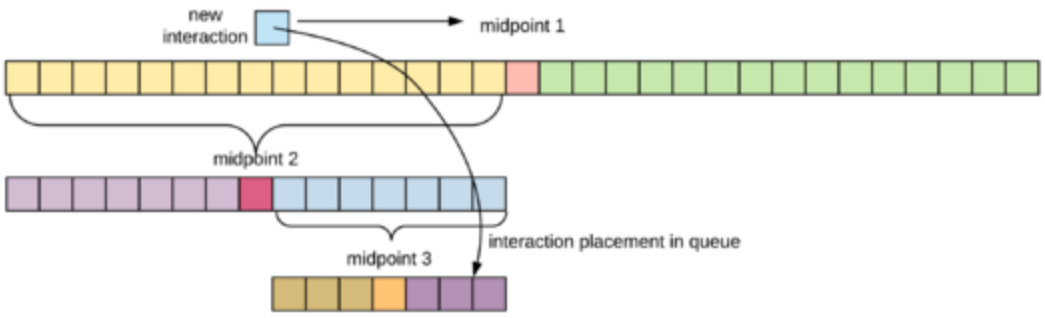
How Interactions are Sorted within the Queue

As each interaction comes in, it is scored, and then assessed relative to the interaction at the midpoint of the existing array of interactions. Should GPR route it before or after the mid-point interaction?

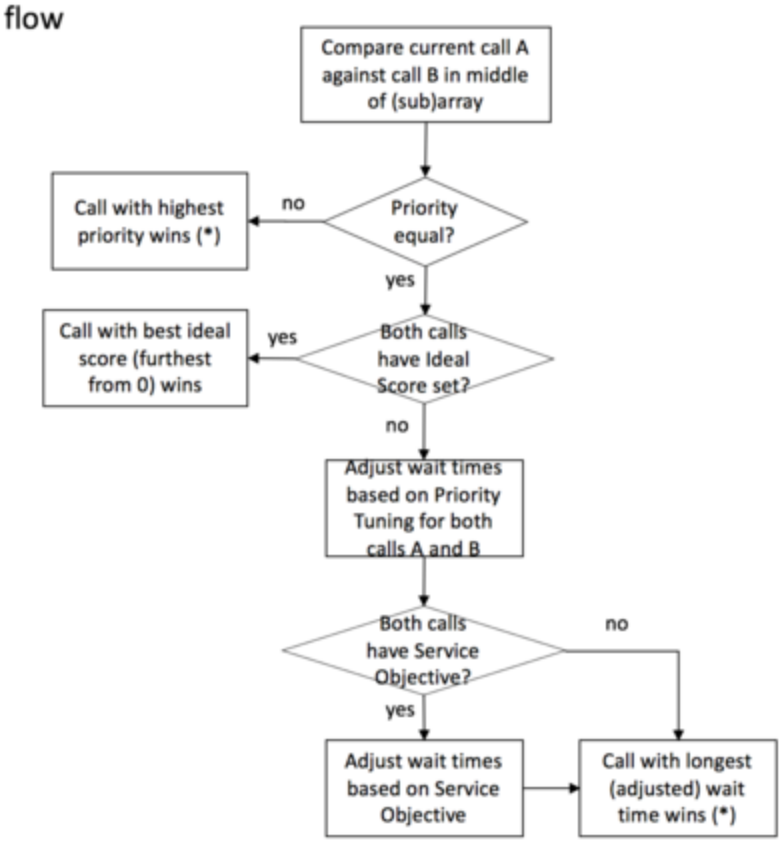
> **Important**
>
> The order in which interactions are prioritized is called an *array* here. This is not equivalent to a queue. These interactions might be from multiple queues, each of which is submitting interactions for URS sorting and routing.

After this decision, URS compares the new interaction against the midpoint within the selected region. Each time URS evaluates the interaction, it is assigned to a smaller region with the total array, always relative to the midpoint of the previous region.

> **Important**
>
> The order in which interactions are prioritized is called an *array* here. This is not equivalent to a queue. These interactions might be from multiple queues, each of which is submitting interactions for URS sorting and routing.



The sorting decision tree:



Example 1

Three calls arrive at a contact center:

- C1 - priority = 1, agent score = 0.3, timestamp = 0:00, URS ID = 1
- C2 - priority = 1, agent score = n/a, timestamp = 0:05, URS ID = 2
- C3 - priority = 1, agent score = 0.6, timestamp = 0:10, URS ID = 3

1. C1 arrives first and is placed into the empty array.
2. C2 arrives. URS compares it with the middle (in this case, only) call in the array, C1.
   The priority is equal and, because C2 has no agent score, URS moves to the next decision criterion.
3. C2 has a shorter wait time, so is put behind C1.
   With only two calls in the array, no further comparison is needed.
4. C3 arrives. URS compares it against the "middle" entry of the array, C1.
   The priority is equal. C3 has better score (further from 0), so URS puts it in front of C1.

Outcome: C3, C1, C2 (the example assumes that interactions are taken from the left end of the array)

Example 2

Five calls arrive at a contact center and are placed in either the Predictive Routing queue or a

conventional queue:

- C1 - priority = 1, agent score = n/a, timestamp = 0:00, URS ID = 1
- C2 - priority = 1, agent score = 0.5, timestamp = 0:05, URS ID = 2
- C3 - priority = 1, agent score = n/a, timestamp = 0:10, URS ID = 3
- C4 - priority = 1, agent score = 0.75, timestamp = 0:15, URS ID = 4
- C5 - priority = 1, agent score = 0.95, timestamp = 0:20, URS ID = 5

1. C1 arrives first. URS places it into the empty array.
2. C2 arrives. URS compares it with the middle (in this case, only) call in the array, C1.
   The priority is equal and, because C1 has no agent score, URS moves to the next decision criterion.
3. C2 has a shorter wait time, so is put behind C1. (In this example,
   Current order: C1 C2 (the example assumes that interactions are taken from the left end of the array)

   With only two calls in the array, no further comparison is needed.
4. C3 arrives. URS compares it against the "middle" entry of the array, C2.
   The priority is equal and, because C3 has no agent score, URS moves to the next decision criterion.
5. C3 has a shorter wait time, so is put behind C2.
   Current order: C1 C2 C3
6. C4 arrives. URS compares it against the middle entry of the array, C2.
   The priority is equal. C4 has a better score (further from 0), so URS places it before C2.
7. Now URS must determine whether C4 should be before or after C1, which is also before C2.
   The priority is equal and, because C3 has no agent score, URS moves to the next decision criterion.
8. C4 has a shorter wait time (a more recent timestamp), so URS places it behind C1.
   Current order: C1 C4 C2 C3
9. C5 arrives. URS compares it against the "middle" entry of the array, C2.
   The priority is equal. C5 has a better score (further from 0), so URS places it before C2.
10. Now URS must determine whether C5 should be before or after C4, the "middle" call in the section of the array before C2.
    The priority is equal. C5 has a better score, so URS places it before C4.
11. Now URS must determine whether C5 should be before or after C1.
12. C5 has a shorter wait time, so URS places it behind C1.
    Final order: C1 C5 C4 C2 C3

Using Agent Hold-Out

Agent hold-out enables you to have an interaction wait a specified time, even when an agent has become available, if the available agent is has a low score for the interaction and there is a chance a better-matched agent might become available within the configured time window. The interaction flow is as follows:

1. URS calls the IsAgentScoreGood subroutine, which determines whether any of the available agents meet the threshold for handling the interaction.

2. If available agents have low scores for this interaction and the interaction spent only a short time in the queue, URS waits for a better agent to become ready.
3. The minimum acceptable score required for an agent for the interaction is gradually reduced, so if no higher-scored agent becomes available, the lower-scored agent might finally be given the interaction.

After that determination occurs, the remainder of the flow is the same as that given in the agent-surplus flow above. Use the relevant Predictive_Route_DataCfg Transaction List Object configuration options to set up the priority increments.

Dynamic Interaction Priority Increments

To avoid having interactions lingering in a queue for an excessive amount of time, URS can trigger an escalation in interaction priority after a time delay that you set. To speed up interaction handling, you can incrementally relax the minimum skill level required for agents to handle the interaction or expand the pool of agents to consider.

Each time a routing strategy tries to route an interactions, it calls the ActivatePredictiveRouting subroutine. After each failed routing attempt, the strategy checks how long the interaction has been waiting in the queue and, if the time in queue is above a certain threshold, it routes the interaction to the next available agent, no matter their score for the interaction.

Use the relevant Predictive_Route_DataCfg Transaction List Object configuration options to set up the priority increments.

Using GPR with agent reservation

When your Genesys environment contains multiple URS nodes, they might compete to distribute interactions to the same pool of agents. URS uses *agent reservation* functionality to resolve which interaction should be routed to a particular agent. For details, see the **agent_reservation** option in the *Universal Routing 8.1 Reference Manual*.

Note: If you are using Service Objective, or Prediction/What-if routing, consult Genesys Customer Care for how to agent reservation works in these scenarios.

For agent reservation, a URS node sends an agent reservation request to a dedicated SIP Server/T-Server application with three extensions: priority, ar-priority-1, and ar-priority-2. The value for the extensions depends on whether this is a GPR interaction and what value you have set for the URS **automatic_ideal_agent** option.

For GPR interactions:

- **priority**: The current interaction priority.
- **ar-priority-1**: The score assigned by URS to the agent for this interaction calculated using the following formula: (100 - (*max-score*> -)). The value can be negative.
- **ar-priority-2**: The time the interaction spent in the strategy Target block, in milliseconds.

For non-GPR interactions when the URS **automatic_ideal_agent** option is set to a positive value (the recommended configuration):

- **priority**: The current interaction priority.
- **ar-priority-1**: The value for this extension is calculated as (100 - automatic_ideal_agent option>). The value can be negative.
- **ar-priority-2**: The time the interaction spent in the strategy Target block, in milliseconds.

For non-GPR interactions when the URS **automatic_ideal_agent** option is set to `false`:

- **priority**: The current interaction priority.
- **ar-priority-1**: The integer part of the time spent by the interaction in the strategy Target block, in seconds.
- **ar-priority-2**: The fractional part of a second of the time spent by the interaction in the strategy Target block, in milliseconds (value range is 0-999).

Example 1

Two URS nodes send concurrent agent reservation requests to SIP Server/T-Server targeting the same agent to handle GPR interactions. When SIP Server/T-Server resolves this race condition, it first compares the interaction priority values (the values of the **priority** Extension key). If they are equal, SIP Server/T-Server compares the values of the **ar-priority-1** keys, which were calculated based on agent scores for the interaction. If those values are also the same, SIP Server/T-Server assigns the interaction waiting longer in the Target block, based on the values of the **ar-priority-2** keys, to the agent. URS fails the routing attempt for the other interaction and the routing strategy handles rerouting it. GPR reports the failed routing attempt using the reporting key **gpmResult=14**.

Example 2a

Two URS nodes have the **automatic_ideal_agent** configuration option set to **45**. Predictive Routing has the ***max-score*** set to **100**. URS node 1 targets a GPR interaction to an agent with an **ar-priority-1** score of **60** for the interacrion. Concurrently, URS node 2 targets a non-GPR interaction to the same agent. The agent has an **ar-priority-1** score of **55** for that interaction. Based on these scores, the SIP Server/T-Server handling agent reservation assigns the GPR interaction from URS node 1 to the agent.

Example 2b

This example has a scenario similar to Example 2a, but the ***max-score*** and **automatic_ideal_agent** have different values.

Two URS nodes have the **automatic_ideal_agent** option set to **4500**. The Predictive Routing ***max-score*** option is set to **10000**. URS node 1 targets a GPR interaction to an agent for which the GPR Core Platform has assigned a score of **6000**. According to the formula for the calculation of ar-priority-1 (see the beginning of this section), the agent's adjusted value is **(100 - (10000 - 6000))=-3900**. Concurrently, URS node 2 targets a non-GPR interaction to the same agent, who has an ar-priority-1 value of **(100 - 4500)** = **-4400** for that non-GPR interaction. Based on these values, the SIP Server/T-Server handling agent reservation assigns the GPR interaction from URS node 1 to the agent.

Example 3

Two URS nodes have the **automatic_ideal_agent** option set to **false**. Predictive Routing has the ***max-score*** set to **100**. URS node 1 targets a GPR interaction, Interaction 1, to an agent for which the GPR Core Platform has assigned a score of **60**. Concurrently, URS node 2 targets a non-GPR

interaction, Interaction 2, to the to the same agent, who has the same ar-priority-1 value of **60** for that non-GPR interaction. Interaction 1 spent 67.3 seconds in the strategy Target block while Interaction 2 spent 180.5 seconds there. The SIP Server/T-Server handling agent reservation receives the requests: one from URS node 1 with **ar-priority-1=60**, **ar-priority-2=67300** and the other from URS node 2 with **ar-priority-1=180**, **ar-priority-2=500**. As a result, Interaction 2 is routed to the agent.

Configure A/B comparison test ratios

To test how GPR handles routing on your queues compared with skills-based routing, you can configure an A/B (comparison) test. GPR handles a specified percentage of interactions (GPR on) and the rest are routed by your original routing method (GPR off).

To start using comparison testing, set the ***prr-mode*** option to **ab-test-time-sliced**. This turns on comparison testing mode.

Start A/B testing

Genesys recommends the following sequence:

1. Start with a fourteen-day, 50/50, hour-on hour-off comparison test in your test environment.
2. Based on a satisfactory result, turn GPR on for all interactions in your production environment.
3. (Optional) - To check that GPR continues to generate improvement, run a non-50/50 comparison test in your production environment using a ratio such as 90/10 or 80/20. Such a split enables you to retain most of the benefit from using GPR while verifying GPR performance compared with traditional routing.

Instructions for configuring both 50/50 and non-50/50 time splits follow.

Notes on how A/B testing works

- When configuring A/B test periods, Genesys recommends that the time slices should be no shorter than 3600 seconds (one hour) and, if you use longer time slices, that they should be multiples of one hour.
- To ensure that the A and B test slices are comparable, GPR alternates which routing method is on during a specific time slice, both daily and weekly.
  - **Example 1** - With an hour-on, hour-off 50/50 test cycle (the default setting), Monday 8-9 uses A, Tuesday 8-9 uses B. The slices are also switched weekly, so that in week 1 Monday 8-9 is A and in week 2, Monday 8-9 is B. By the end of a two-week comparison period, each routing method has been used for each time-slice period.
  - **Example 2** - With an eight hours on, two hours off 80/20 test cycle (as explained in the "Configure a non-50/50 comparison test time split" section, below), Monday midnight - 8 am uses A, 8-10 uses B, and so on. In this pattern, the times of day routed using each method shift automatically. This shift produces a full cycle that uses each routing method for each time-slice period. Assuming your environment is active twenty-four hours, a single full cycle takes six days. You must then adjust the length of the comparison test depending on the nature of your environment and the KPI you specify.
- A comparison test should always run at least fourteen days. In most cases, you must run the test significantly longer to ensure that you produce consistent results. The length of a comparison test should take the following factors into account:

- **KPI type** - A KPI such as AHT, which returns an immediate result, requires a shorter test period than a KPI, such as first contact resolution, that takes time before the KPI outcome for an interaction is available.
- **Time ratio configured** - Non-50/50 time splits require longer comparison tests. The further from 50/50, the longer the comparison test should be to achieve solid results.
- **Improvement with GPR** - The stronger the percent improvement using GPR, the shorter the comparison test period can be.
- **Example** - In a test of AHT showing a 3% benefit using GPR, Genesys recommends that you run a test with an 80/20 split for 30 days, and a 90/10 split for 44 days.

Note: A/B comparison testing is based on time-on/time-off segments. The number of calls routed by each method might not exactly match the specified percentage because the number of calls coming into the queue might vary throughout the comparison test period.

Configure a 50/50 comparison test time split

For a 50/50 comparison test split, set the ***ab-test-time-slice*** option to a positive value. This is the time, in seconds, for each routing method to run (the time slice).

If you do not configure this option, A/B test periods are hourly by default. For more information, see the complete description of the ***ab-test-time-slice*** option.

- Genesys recommends that your time slice be at least 3600 seconds in a production environment.
- For robust results, run a 50/50 comparison test time split for at least 14 days with the times slices no shorter than one hour on, one hour off (this is the default setting).

Formula Used

GPR uses the following formula to determine the GPR Mode for a particular call when you configure a 50/50 test time split:

```
a = varStartTS mod (varABTestTimeSlice * 2)
if a = varABTestTimeSlice = GPR ON
```

where

- `varABTestTimeSlice` refers to the value set for the option, **ab-test-time-slice**.
- `varStartTS` refers to the date and time at which the interaction began as a Coordinated Universal Time (UTC) value. The UTC time is formatted as Unix time value.
  For example, if the call start time is *Thursday, 5 May 2022 5:30:00 AM GMT+05:30*, it is converted to UTC time as *Thursday, 5 May 2022 12:00:00 AM*. Then the corresponding Unix timestamp is identified, in this case, *1651708800* and will be used as the `varStartTS` value.

Using this start timestamp and when the **ab-test-time-slice** option is set to 86400 seconds (24 hours), the result for the calculation will enable GPR for all interactions coming on this day. For the next day, GPR will be turned off for the whole day.

Default Configuration

By default, the ***ab-test-time-slice*** option is set to zero. This will toggle GPR on and off every hour. The first hour, GPR will be enabled and in the second hour, it will be disabled and in the third hour, GPR will be re-enabled.

The formula used in the default configuration is: `varABTestTSUsePredictive = (dayOfyear % 2 + hourOfDay % 2 ) % 2` where `dayOfyear` indicates the number of days elapsed since January 1 of that year and the interaction date and `hourOfDay` indicates the hours elapsed since midnight.

Consider an example scenario where the **timestamp** value is *1653052118*, the **dayofyear** value is *140*, and the **hourofday** value is *13*, applying the formula gives the values:

- dayofyear%2 = 0
- hourofday%2 = 1
- varABTestTSUsePredictive = 1

In this scenario, GPR is enabled.

Similarly, consider an example scenario where the **timestamp** value is *1653049472*, the **dayofyear** value is *140*, and the **hourofday** value is *12*, applying the formula gives the values:

- dayofyear%2 = 0
- hourofday%2 = 0
- varABTestTSUsePredictive = 0

In this scenario, GPR is disabled.

Configure a non-50/50 comparison test time split

If you would like to move GPR into production but monitor the benefit at the same time, Genesys recommends using a non-50/50 split. To set the comparison test time periods to a non-50/50 split, specify values that produce the desired time split in the following two configuration options:

- ***ab-test-gpr-on-period***
- ***ab-test-gpr-off-period***

If you leave these options at the default settings, the comparison test defaults to a 50/50 split, with interactions routed half the time using GPR and half using skills-based routing. The on/off periods (time slices) are defined by the value set in the ***ab-test-time-slice*** option.

Example configuration for a 90/10 split

- **ab-test-gpr-on-period** - 60*60*9 (9 hours ON)
- **ab-test-gpr-off-period** - 60*60*1 (1 hour OFF)

Example configuration for an 80/20 split

- **ab-test-gpr-on-period** - 60*60*8 (8 hours ON)
- **ab-test-gpr-off-period** - 60*60*2 (2 hour OFF)

To get high-quality results from comparison testing, the test must run long enough to smooth out any unusual occurrences or statistical anomalies in the data. The further the split is set from a 50/50 ratio, the longer the comparison test period should be to achieve a high-quality test.

Formula Used

GPR uses the following formula to determine the GPR Mode for a particular call when you configure a non-50/50 test time split:

```
varStartTS mod (GPR_ON_PERIOD + GPR_OFF_PERIOD)
if a = GPR_OFF_PERIOD = GPR ON
```

where

- `GPR_ON_PERIOD` refers to the value set for the option, **ab-test-gpr-on-period**
- `GPR_OFF_PERIOD` refers to the value set for the option, **ab-test-gpr-off-period**
- `varStartTS` refers to the date and time at which the interaction began as a Coordinated Universal Time (UTC) value. The UTC time is formatted as Unix time value.
  For example, if the call start time is *Thursday, 5 May 2022 5:30:00 AM GMT+05:30*, it is converted to UTC time as *Thursday, 5 May 2022 12:00:00 AM*. Then the corresponding Unix timestamp is identified, in this case, *1651708800* and will be used as the `varStartTS` value.

Using this start timestamp and when the **ab-test-gpr-on-period** is set to *28800* seconds (8 hours) and **ab-test-gpr-off-period** to *7200* seconds (2 hours), applying the formula will result in disabling GPR for this interaction.

How Does GPR Score Agents?

Contents

GPR scores agents based on the data uploaded to the Agent and Customer Profiles or, if you are using the GPR API, on data passed in the API score request as part of the context parameter. If both are present, data from the API request takes priority over data from the Agent and Customer Profiles.

This topic explains how GPR handles various scoring scenarios, depending on your environment and your configuration settings.

Related documentation:

-

Default Agent Scores

If an agent belongs to the target Agent Group but GPR does not score the agent, the isAgentScoreGood and ScoreIdealAgent subroutines assign a score for that agent according to the value set for the default-agent-score configuration option.

For agents who have a default score assigned, the following KVPs reflect that value:

- gpmAgentScore, which records the value specified in the **default-agent-score** option if the agent who handled the interaction had the default score. If the the AICS scoring engine calculated a score for the agent, gpmAgentScore reports the calculated score value.
- gpmDefaultAgentScore, which records the value specified in the **default-agent-score** option.
- gpmDefaultScoreUsed, which indicates whether the selected agent was assigned the default score.
- gpmDefaultScoredAgents, which records the number of agents assigned the default agent score.

Example 1

Agents A and B log in after the scoring request is made and are each assigned the default score, which is 40. Agent A receives the interaction. The related KVPs have the following values:

- gpmAgentScore = 40
- gpmDefaultAgentScore = 40
- gpmDefaultScoreUsed = 1
- gpmDefaultScoredAgents = 2

Example 2

GPR assigns Agent C a score of 80. The default score is 40. No agents are assigned the default score.

The related KVPs have the following values:

- gpmAgentScore = 80
- gpmDefaultAgentScore = 40
- gpmDefaultScoreUsed = 0
- gpmDefaultScoredAgents = 0

Score Adjustment

The GPR subroutines enable you to adjust agent scores using an Occupancy factor when URS sorts them. You control the agent occupancy setting in the agent-occupancy-factor configuration option. Scores adjusted using an agent occupancy factor are recorded in the gpmAdjustedAgentScore KVP.

Example 1

GPR returns a score of 80 for Agent A. The **agent-occupancy-factor** option value is `0.5`. If this agent selected to receive the interaction, the agent score KVPs have the following values:

- gpmAdjustedAgentScore = 40
- gpmAgentScore = 80

Example 2

- Agent A is available and has an occupancy of 80% and a score of 75
- Agent B is available and has an occupancy of 60% and a score of 70

The following settings are configured:

- agent-occupancy-factor = 0.5
- use-agent-occupancy = true
- agent-occupancy-threshold = 70%

Agent A

- gpmAdjustedAgentScore = 40
- gpmAgentScore = 80

Agent B

- gpmAdjustedAgentScore = 75
- gpmAgentScore = 75

As a result, the call is routed to Agent B

> **Important**
>
> This adjusted score is used only for sorting the agent scores in the ScoreIdealAgent subroutine. The adjusted agent score is *not* used in the isAgentScoreGood subroutine to compare the agent score with the configured threshold. The actual returned score is used.

Threshold Scores

To implement the agent holdout feature, GPR checks the score returned for the agent against the threshold value configured in the score-base-threshold option. URS calls the isAgentScoreGood subroutine to suppress routing to an agent who is in ready state if this agent does not provide an acceptable match for the interaction. This is used in conjunction with relaxation thresholds to target better-matched agents preferentially, expanding the pool of agents if the best-matched agents are unavailable.

- See Agent Holdout Options for the complete list of options used for agent holdout and threshold settings.

Score Relaxation Timeouts

By design, URS checks the threshold relaxation ("awakens") at two-second intervals. As a result, the minimum *real-world* value for threshold-relaxation-timeout is 2 because the threshold relaxation is checked only every two seconds. Even though the default value for the **threshold-relaxation-timeout** option is 1, URS applies the threshold relaxation only at two-second intervals.

When the **initial-threshold-timeout** value has elapsed, the minimum score required to allow an agent to handle the interaction is reduced by the configured relaxation step. This relaxation step can be applied multiple times, depending on the option settings you specify.

Example 1

GPR returns a score of 50 for Agent A. The threshold and relaxation options have the following values:

- score-based-threshold = 55
- initial-threshold-timeout = 2
- threshold-relaxation-timeout = 4
- threshold-relaxation-step = 4

Agent A is selected after 6 seconds. The related KVPs have the following values:

- gpmAgentScore =50
- gpmInitialScoreThreshold = 55

- gpmFinalScoreThreshold = 47

URS attempts for Agent A	Threshold Value	Result
Interaction queued and scoring completed in the same second	55 (initial threshold value)	agent score (50)
after 2 seconds	51 (first relaxation applied, 55-4)	agent score (50)
after 4 seconds	51 (no change from previous step)	agent score (50)
after 6 seconds	47 (second relaxation applied, 51-4)	agent score (50) > threshold (47); interaction routed to agent

Example 2

GPR returns a score of 30 for Agent B. The threshold and relaxation options have the following values:

- score-based-threshold = 40
- initial-threshold-timeout = 5
- threshold-relaxation-timeout = 2
- threshold-relaxation-step = 5

Agent A is selected after 8 seconds. The related KVPs have the following values:

- gpmAgentScore =30
- gpmInitialScoreThreshold = 40
- gpmFinalScoreThreshold = 30

URS attempts for Agent B	Threshold Value	Result
Interaction queued and scoring completed in the same second	40 (initial threshold value)	agent score (30)
after 2 seconds	40 (no change from previous step)	agent score (30)
after 4 seconds	40 (no change from previous step)	agent score (30)
after 6 seconds (initial timeout is 5 seconds, but relaxation is applied only when URS awakens)	35 (first relaxation applied, 40-5)	agent score (30)
after 8 seconds	30 (second relaxation applied, 35-5)	agent score (30) = threshold (30); interaction routed to agent

Example 3

GPR returns a score of 35 for Agent C. The threshold and relaxation options have the following values:

- score-based-threshold = 40

- initial-threshold-timeout = 5
- threshold-relaxation-timeout = 1 (no value specified, default value used)
- threshold-relaxation-step = 1 (no value specified, default value used)

Agent C is selected after 10 seconds. The related KVPs have the following values:

- gpmAgentScore =35
- gpmInitialScoreThreshold = 40
- gpmFinalScoreThreshold = 34

URS attempts for Agent C	Threshold Value	Result
Interaction queued and scoring completed in the same second	40 (initial threshold value)	agent score(35)
after 2 seconds	40 (no change from previous step)	agent score (35)
after 4 seconds	40 (no change from previous step)	agent score (35)
after 6 seconds (initial timeout is 5 seconds, but relaxation is applied only when URS awakens)	38 first and second relaxation applied: • first relaxation after initial 5 seconds • second relaxation after next 1 second	agent score (35)
after 8 seconds	36 (third and fourth relaxations applied, 38 - 2)	agent score (35)
after 10 seconds	34 (fifth and sixth relaxations applied, 36 - 2)	agent score (35) > threshold (34); interaction routed to agent

How the Availability Status of Agents in the Target Agent Group is Determined

URS provides agent availability information to GPR. It checks with Stat Server on the agent login status of the specified target group before making the scoring request, and adds the list of matching agents in the request field 'action_filters'.

The GetActionFilters subroutine reads the login statuses specified to be available for routing from the login-status-expression configuration option.

- To use this functionality, set the value of the use-action-filters configuration option to `false` (the default value is `true`).

Sample scoring request showing a list of agents employee IDs in the field `action_filters`, where

POC0x strings indicate IDs of agents with a required login status:

```
{
"token":"",
"format_as_map":"true",
"context_id":"3600",
"log_request":"true",
"action_filters":"EMPLOYEE_ID in [\"POC01\",\"POC02\",\"POC03\",\"POC04\"]",
"context":
{
    "PR_TYPE":"Gold",
   "PR_LANG":"French"
}
}
```

Important

- This architecture increases the load on URS by approximately 15%. Use the Sizing Guide to verify that you have sufficient URS bandwidth available.
- Only alphanumeric characters, spaces, and underscores are supported in the names of Stat Server `Application` objects. Names including other special characters cause a malformed scoring request.
- The EMPLOYEE_ID field should not contain the parenthesis characters (or). When the ***use-action-filters*** option is set to `false` and a scoring request contains an agent `EMPLOYEE_ID` value that includes (or), GPR returns an error.

Data Loader monitoring and logging

Contents

You can monitor Data Loader status using the log events that Message Server sends to Solution Control Interface.

Related documentation:

-

Data Loader Log Events

Data Loader writes log data to Message Server using Genesys Management Framework centralized logging. The details of where logs are kept and what messages are sent are configured using the Data Loader **log** Section configuration options.

Data Loader provides the following Standard-, Info-, and Trace-level log events:

107-60119	107-60120	107-60400	107-60401	107-60606	107-60607	107-60608
107-60609	107-60610	107-60620	107-60702	107-60706	107-60707	107-60801
107-60802	107-60803	107-60804	107-60805	107-60806	107-60807	107-60808
107-60809	107-60810	107-60811	107-60812	107-60813		

107-60119

Level	Standard
Text	data_loader_name> terminated, reason>
Attributes	data_loader_name> - The name of the Data Loader `Application` and other relevant information. reason> - An explanation for the termination.
Description	Data Loader has quit unexpectedly on account of a critical configuration issue. The reason> information should direct you to the source of the problem.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message.
Actions	Review the reason for the termination and correct the issue before trying to upload data.

107-60120

Level	Standard
Text	Failed to connect to authenticate endpoint>, response: message>
Attributes	endpoint> - The endpoint for the GPR Core Platform authentication service. response> - An informational message indicating the reason the authentication attempt failed.
Description	Data Loader authentication to the GPR Core Platform has failed.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message.
Actions	Verify that you have correctly configured the URL for Data Loader to connect to the Platform and that the Data Loader account is correctly configured with a valid password.

107-60400

Level	Standard
Text	error... error_message>
Attributes	error_message> - An informational message.
Description	Data Loader was unable to push an Agent Profile batch update. If you receive this error message, Data Loader continues to operate normally.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message.
Actions	No action needed.

107-60401

Level	Standard
Text	The log event text is one of the following: • The following Agent Groups are not found in Configuration Server. Please verify the correct names for these Agent Groups: agent_group_names> in Data Loader option: option_name>

	• There are no agents present in Configuration Server with the following skills. Please verify the correct names for these skills: skill_names> in Data Loader option: option_name>
Attributes	agent_group_names> or skill_names> - The name of the Agent Group or skill you entered. option_name> - The option where you entered the Agent Group or skill name.
Description	This log number has two possible messages indicating that a value for the ***include-skills*** or ***include-groups*** configuration option does not correspond to a configured skill or Agent Group. This log event is informational and does not indicate any issue with Data Loader.
Alarm Advisory	No alarm required.
Actions	Verify the names of the Agent Groups or skills and make corrections as necessary.

107-60606

Level	Standard
Text	DATASET_UPLOAD_FAILURE: Datasets upload errors: error_code>
Attributes	error_code> - An error message or list of error messages indicating an issue with dataset processing.
Description	This log event could provide any of a number of issues related to dataset processing: configuration issues, Data Loader issues, and GPR Core Platform issues. The error message indicates the nature of the problem.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message. The cancel event for this condition is 107-60607.
Actions	If a GPR Core Platform error caused the issue, rerun the data upload manually by setting the ***upload-dataset*** option to `false`, waiting for one minute, then setting the option value back to `true`. Alternatively, you can wait for the next scheduled upload of the data, as configured in the ***update-period*** option.

107-60607

Level	Standard
Text	Datasets upload errors resolved!
Attributes	none
Description	An informational message generated when a data upload is successful after an unsuccessful attempt. Confirms that whatever issue caused the previous upload failure has been resolved.
Alarm Advisory	This is the cancel event for 107-60606.
Actions	No action required

107-60608

Level	Standard
Text	Platform service account password is about to expire in number_of_days> days.
Attributes	number_of_days> - The number of days the current password remains valid.
Description	An informational message.
Alarm Advisory	No alarm required.
Actions	Change your password within the specified period.

107-60609

Level	Standard
Text	Platform service account password updated successfully
Attributes	none
Description	An informational message indicating that the password for the Data Loader SERVICE account was successfully updated.
Alarm Advisory	No alarm required.
Actions	No action required.

107-60610

Level	Standard

Text	Platform service account password update failed.
Attributes	None
Description	A notification that the password for the Data Loader SERVICE account was not correctly updated.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message.
Actions	Restart Data Loader. If this issue persists, contact Genesys Customer Care.

107-60620

Level	Standard
Text	agent/customer_profile_name> profile schema doesn't match the schema defined in your schema configuration; problem_description>
Attributes	agent/customer_profile_name> - The name of the Agent Profile dataset or Customer Profile dataset that has inconsistencies. problem_description> - Explains what the inconsistencies are.
Description	The data you are uploading to the Agent Profile dataset or Customer Profile dataset does not correspond with the configuration specified for that dataset in the Data Loader `Application` object options.
Alarm Advisory	No alarm required.
Actions	Make the Profile schema consistent with the configuration specified on the **Options** tab of the Data Loader `Application` object.

107-60702

Level	Standard
Text	CONFSERV_REPEATABLE_SWITCHOVER_EVENTS: error_message>
Attributes	error_message> - A message explaining the nature of the issue with Configuration Server.
Description	Configuration Server has experienced multiple switchovers between the primary and backup instances during the period specified in the ***confserv-monitoring-reconnect-min*** option.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error

	message.
Actions	Check whether Configuration Server is experiencing some issue causing the repeated switchovers.

107-60706

Level	Standard
Text	CONFSERV_DISCONNECTING_EVENT: error_message>
Attributes	error_message> - A message explaining the nature of the issue with Configuration Server.
Description	Configuration Server is losing connection with Data Loader. The number of times the connection is lost before this alarm is triggered is set in the ***confserv-monitoring-reconnect-count*** option.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message. The cancel event for this error condition is 107-60707.
Actions	Check Configuration Server to determine the reason for this issue.

107-60707

Level	Standard
Text	CONFSERV_CONNECTING_EVENT: informational_message>
Attributes	informational_message> - A message updating the Configuration Server status.
Description	The Configuration Server connection with Data Loader has been restored.
Alarm Advisory	This is the cancel event for 107-60706.
Actions	No action required.

107-60801

Level	Standard
Text	ANONYMIZATION_SALT_GENERATION_FAILED: Premise anonymization enabled but salt generation failed with error: error_message>

Attributes	- A explanatory message.
Description	An error occurred causing Data Loader to be unable to generate the salt for anonymization. This prevents on-premise anonymization.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	Review the error message text for information about the cause of this issue.

107-60802

Level	Standard
Text	ANONYMIZATION_SALT_UPDATED_BY_USER: Anonymization salt modified by the user in modified_application_object>
Attributes	modified_application_object> - Identifies the `Application` object in which a user changed the salt value.
Description	A user has changed the value for the '*salt* option in the specified `Application` object. This message is informational. Data Loader automatically resets the salt to the original value.
Alarm Advisory	Genesys recommends that you set an alarm for this event. The clearance event for this warning message is 107-60803.
Actions	Remind the user who made the change that the salt value should not be edited. If the salt is changed, your predictors become unusable for scoring and routing.

107-60803

Level	Standard
Text	ANONYMIZATION_SALT_RESTORED: Modified anonymization salt restored successfully in modified_application_object>
Attributes	modified_application_object> - Identifies the `Application` object in which Data Loader restored the salt value.
Description	An informational message confirming that Data Loader successfully restored the modified anonymization salt.
Alarm Advisory	This is the clearance event for 107-60802.
Actions	No action needed.

107-60804

Level	Standard
Text	ANONYMIZATION_SALT_CORRUPTED: Salt corrupted by user in config! Unable to recover
Attributes	No attributes.
Description	This error message indicates that a user has changed the **salt** value in such a way that Data Loader is unable to restore the original value. On-premise anonymization requires a new salt, which is incompatible with that used for your current datasets and predictors. For recovery instructions, see Data anonymization.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	For recovery instructions, see Data anonymization.

107-60805

Level	Standard
Text	SUBROUTINES_TRAN_LIST_NOT_FOUND: Subroutines transaction list Predictive_Route_DataCfg not found
Attributes	No attributes.
Description	The Predictive_Route_DataCfg `Transaction List` object does not exist in your configuration environment. This configuration object contains options required to use Predictive Routing.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	Create the Predictive_Route_DataCfg `Transaction List` object in the configuration environment for your Tenant.

107-60806

Level	Standard
Text	SUBROUTINES_TRAN_LIST_DELETED: Subroutines transaction list Predictive_Route_DataCfg deleted by user! Unable to restore anonymization salt
Attributes	No attributes.
Description	A warning message indicating that a user has deleted the Predictive_Route_DataCfg `Transaction`

	`List` object. As a result, Data Loader cannot store the salt value required for the URS Strategy Subroutines to anonymize data on premise.
Alarm Advisory	Genesys recommends that you set an alarm for this log event. 107-60807 is the clearance event for this error.
Actions	Restart Data Loader to restore the anonymization salt value in the **salt** option.

107-60807

Level	Standard
Text	SUBROUTINES_TRAN_LIST_CREATED: Subroutines transaction list Predictive_Route_DataCfg recreated by user! Data loader requires restart
Attributes	No attributes.
Description	A warning message indicating that a user has re-created the Predictive_Route_DataCfg `Transaction List` object. Ensure that this configuration object contains the correct option values for your configuration, and then restart Data Loader.
Alarm Advisory	This is the clearance event for 107-60806.
Actions	Restart Data Loader to restore the anonymization salt value in the **salt** option.

107-60808

Level	Standard
Text	PROFILE_ID_FIELD_NOT_FOUND: Premise anonymization enabled but profile id field not found for profile>
Attributes	profile> - The affected Agent or Customer Profile schema.
Description	Anonymization on premise requires the **anon-agent-id** and **anon-customer-id** options to exist in the Predictive_Route_DataCfg `Transaction List` object.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	Define the missing ID field in the profile> schema.

107-60809

Level	Standard
Text	ANON_PROFILE_ID_CONFIG_FAILED: Premise anonymization enabled but configuring profile id anon field failed for profile>
Attributes	profile> - The affected Agent or Customer Profile schema.
Description	Anonymization is set to happen on-premise but Data Loader cannot specify the correct value for the **anon-agent-id** and **anon-customer-id** options in the Predictive_Route_DataCfg `Transaction List` object.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	Review the error message text for information about the cause of this issue.

107-60810

Level	Standard
Text	PROFILE_ID_ANONYMIZATION_MODIFIED: Profile Id anonymization type modified for profile>
Attributes	profile> - The affected Agent or Customer Profile schema.
Description	The Data Loader **schema-agents-gim** or **schema-customers** setting for the ID field has changes in its anonymization settings, which makes existing datasets and predictors unusable.
Alarm Advisory	Genesys recommends that you configure an alarm to notify you when Data Loader generates this error message.
Actions	Upload the Agent and Customer profiles and datasets once again with the updated schema.

107-60811

Level	Standard
Text	ANON_PROFILE_ID_MODIFIED: Profile Id anonymization type modified by user for profile>
Attributes	profile> - The affected Agent or Customer Profile schema.
Description	A user has manually changed the value for the

	anon-agents-id or **anon-customers-id** option. Data Loader sets these option values automatically. Manual changes to the value for the **anon-agent-id** or **anon-customer-id** options in the Predictive_Route_DataCfg `Transaction List` object cause a discrepancy with the anonymization setting for the ID field specified in **schema-agents-gim** or **schema-customers**.
Alarm Advisory	Genesys recommends that you set an alarm for this log event.
Actions	Data Loader should restore the original option value automatically. Look for log event 107-60812, which is the clearance event for this message, to confirm.

107-60812

Level	Standard
Text	ANON_PROFILE_ID_RESTORED: Profile Id anonymization type restored for profile>
Attributes	profile> - The affected Agent or Customer Profile schema.
Description	Data Loader has corrected the value for the **anon-agents-id** or **anon-customers-id** option after a user manually changed it. Data Loader sets these option values automatically. Manual changes to the value for the **anon-agent-id** or **anon-customer-id** options in the Predictive_Route_DataCfg `Transaction List` object cause a discrepancy with the anonymization setting for the ID field specified in **schema-agents-gim** or **schema-customers**.
Alarm Advisory	This is the clearance event for 107-60811.
Actions	No action needed.

107-60813

Level	Standard
Text	INVALID_PERMISSIONS: Invalid permissions to update configuration parameters in configuration _object>
Attributes	configuration _object> - The affected Data Loader `Application` object or Predictive_Route_DataCfg `Transaction List` object.
Description	Data Loader must be configured to have update privileges for Configuration Server so it can update

	the following: • The password used to access the GPR Core Platform, which is stored in the Data Loader and Predictive_Route_DataCfg configuration objects. • The anonymization salt value created to support anonymization. Data Loader checks whether it has the correct permissions when it starts. If it does not have the required permissions, it quits and generates this log message. The Data Loader log file also records that Data Loader did not start.
Alarm Advisory	You can optionally set an alarm for this log event.
Actions	Grant Data Loader the correct permissions and then restart it.