Predictive Routing Deployment and Operations Guide

Architecture and Security

5/2/2025

Contents

Architecture and Security

This topic presents Genesys Predictive Routing (GPR) architecture, first at a high-level overview, followed by more detailed views of the connections used by AI Core Services (AICS), Agent State Connector (ASC), URS Strategy Subroutines, and Composer Subroutines components.

This topic covers Genesys Predictive Routing architecture, with some additional Genesys components included in the diagrams for completeness. For a full list of required components and versions, refer to System Requirements and Interoperability.

In addition, you need to have adequate data source(s) and construct a well thought-out data pipeline.

- AI Core Services Architecture
- Agent State Connector Architecture
- Subroutines Architecture
- Security and Secure Connections

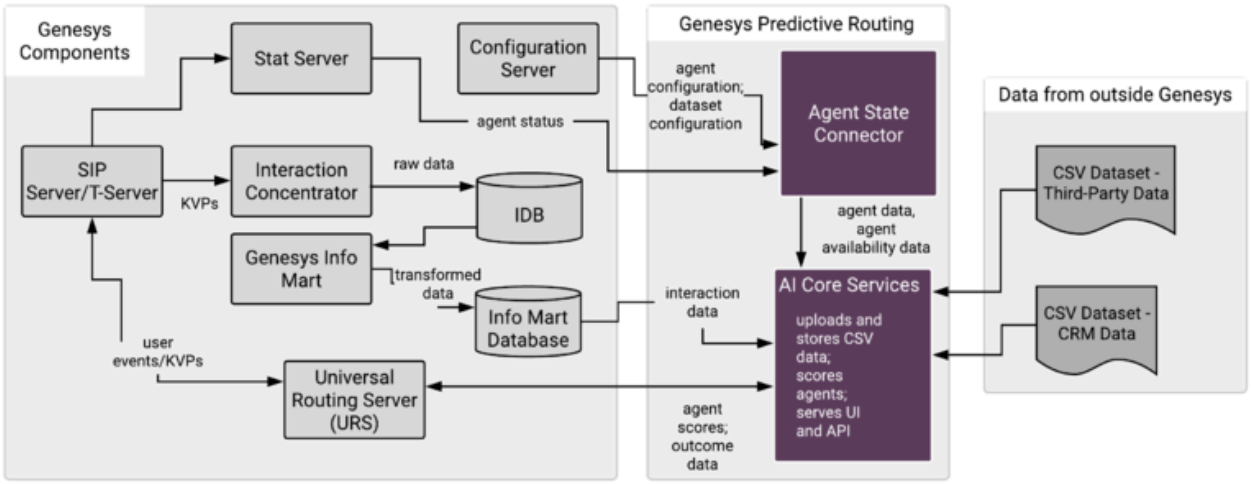
GPR Architecture

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

Important

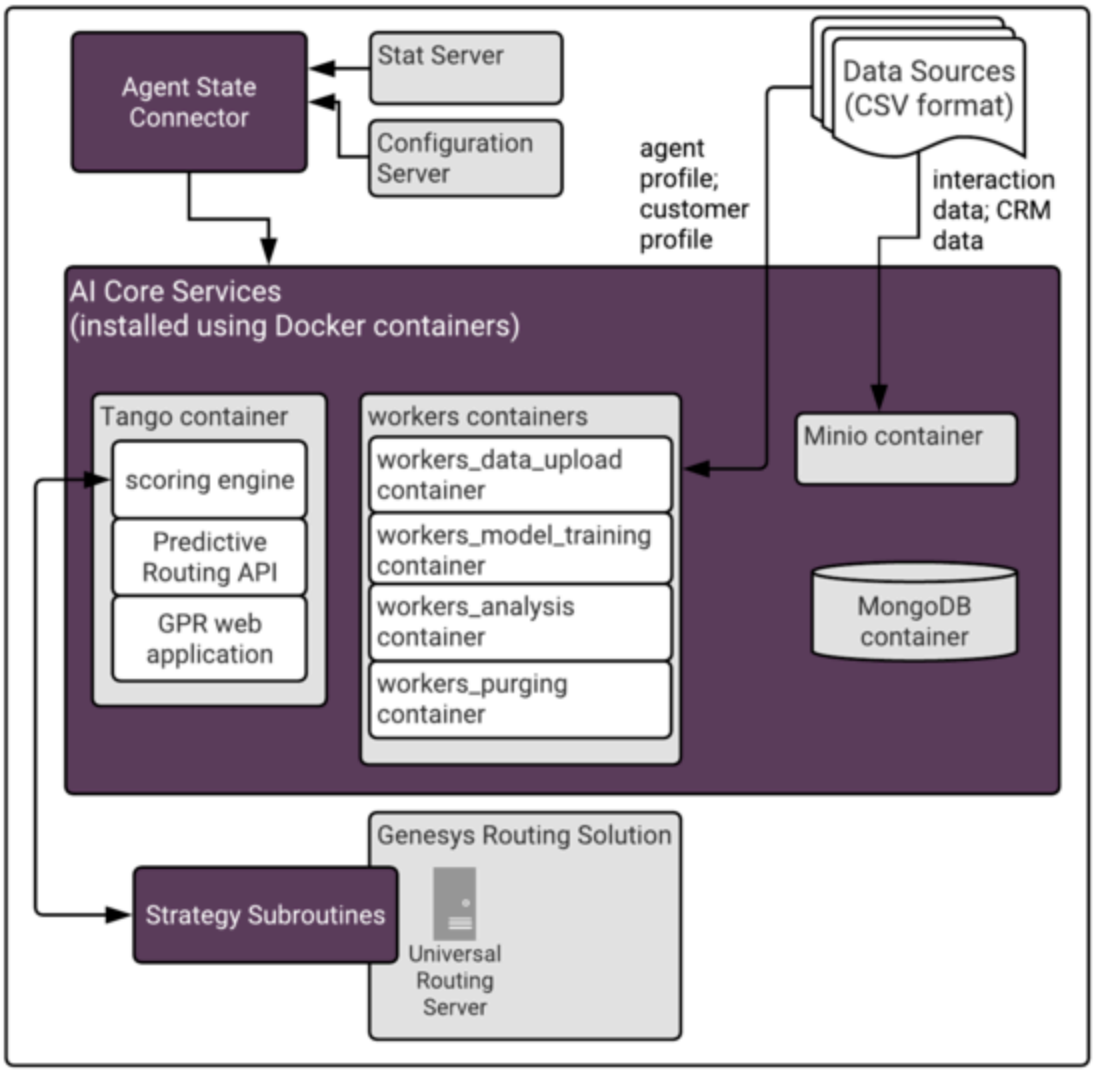
GPR architecture in a high availability (HA) environment is similar to that presented in the diagram except for the following details:

- Your load balancer or Docker container management application distributes traffic to AICS across the configured instances in whatever way your system architects choose. Deploy GPR in an HA Environment explains how to configure AICS if you require HA.
- ASC is a Java application that can be monitored, started, and stopped in Solution Control Interface. It supports a warm-standby high availability architecture.



AI Core Services Architecture

The following diagram shows the AI Core Services (AICS) internal structure and high-level connections to the other GPR components in a single-server deployment.



- See Scaling AI Core Services for information on how to scale each type of container included in the AICS deployment.
- See System Monitoring and Logging for information on how to access the logs for each container.

The Tango Container

Contains the Genesys platform that provides the GPR scoring engine, the Predictive Routing REST API, and the web-based user interface.

Workers Containers

Contain function-specific processes, identifiable by the descriptive container names.

> **Important**
> In earlier releases, the Tango container included the workers listed below. The functionality they provide has not changed. They have been relocated to separate containers to facilitate scaling of specific functional aspects.

- workers_dataset_upload - Uploads agent profile and customer profile data to MongoDB in releases prior to 9.0.015.03. Also uploads Dataset data in releases prior to 9.0.013.01, when the MinIO container was introduced. In release 9.0.015.03 and higher, the MinIO container performs the initial upload for all data, after which the workers_dataset_upload container moves the data from the server-based storage provided by MinIO to MongoDB.
- workers_model_training - Performs Model training jobs.
- workers_analysis - Runs the Feature Analysis, Lift Estimation, and Agent Variance reports.
- workers_purging - Purges your Dataset data.

The MinIO Container

Contains MinIO, which is a high performance, distributed object storage server, designed for large-scale data infrastructure. This container is available in releases 9.0.013.01 and higher, where it improves processing times for the initial Dataset upload. In release 9.0.015.03 and higher, this functionality has been expanded to include Agent Profile and Customer Profile uploads as well.

AICS handles data uploads without any need for you to handle configuration of MinIO. However, if you are interested in more detailed information about this component, see the MinIO web site and documentation.

The MongoDB Container

A highly scalable, highly available no-SQL database which is especially efficient at handling large batches of JSON format data. It also supports fast, efficient queries of that data. Starting in MongoDB 3.2, WiredTiger is the default storage engine for MongoDB.

In high availability (HA) deployments, MongoDB uses *replica sets* split across two data centers. A primary and secondary replica set are located in data center 1 (DC1) and a secondary replica set is located in DC2. All writes go to the primary replica set, from which they are distributed to the secondary replicas. Reads can be directed to either of the secondary sets. You can configure MongoDB to prefer reads from local secondary sets. Cross-site data traffic is required, however, because all writes are directed to the primary MongoDB in DC1 and the data then replicates across sites. Note that sufficient bandwidth will be required for the data replication traffic between data centers.

If the primary server fails, read operations can still continue. Another server can be elected as the primary server to continue write operations. Ideally, an Arbiter node should be set up in a third data center or availability zone. This facilitates the detection of a failed primary node when a data center

becomes inaccessible and the proper election of a primary in the other data center. If there are only two data centers, manual intervention is required to force one of the secondary replicants to become the primary replicant.

- Links to additional information about Mongo DB:
  - WiredTiger Storage Engine
  - https://eladnava.com/deploy-a-highly-available-mongodb-replica-set-on-aws/
  - https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/
  - http://s3.amazonaws.com/info-mongodb-com/MongoDB_Multi_Data_Center.pdf
  - https://stackoverflow.com/questions/43083246/requires-simple-explanation-on-arbiters-role-in-a-givenmongodb-replica-set
  - https://docs.mongodb.com/manual/reference/method/Mongo.setReadPref/

The NGINX Container

> **Important**
>
> - Genesys recommends that you use NGINX only in test (non-production) environments.
> - The NGINX container was removed in release 9.0.015.03.

NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. In addition to its HTTP server capabilities, NGINX is also used as a reverse proxy and load balancer for HTTP, TCP, and UDP traffic.

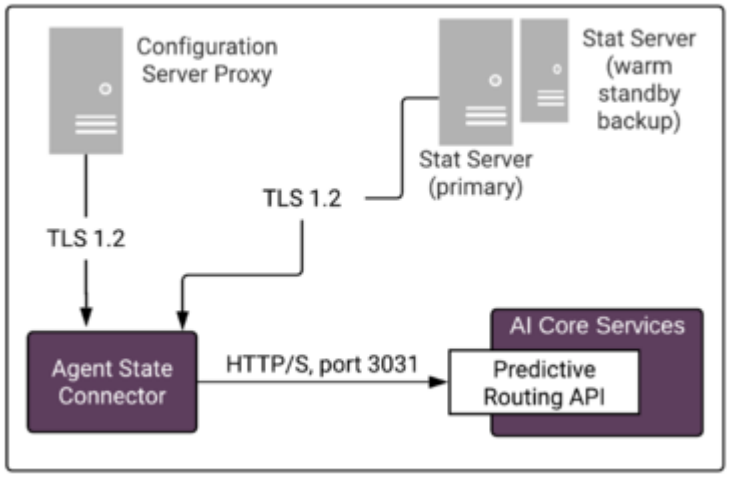
Agent State Connector Architecture

Agent State Connector (ASC) connects to Configuration Server Proxy for agent-related data to be stored in the Agent Profile (agent configuration details, such as a location, languages, skills and skill levels, and so on). You can use this connection to populate the entire Agent Profile or you can upload the initial agent data from a CSV file. In either case, agent data is updated via the connection to Configuration Server. See Configuring Agent Profiles in the *Predictive Routing Help* for additional information and procedures.

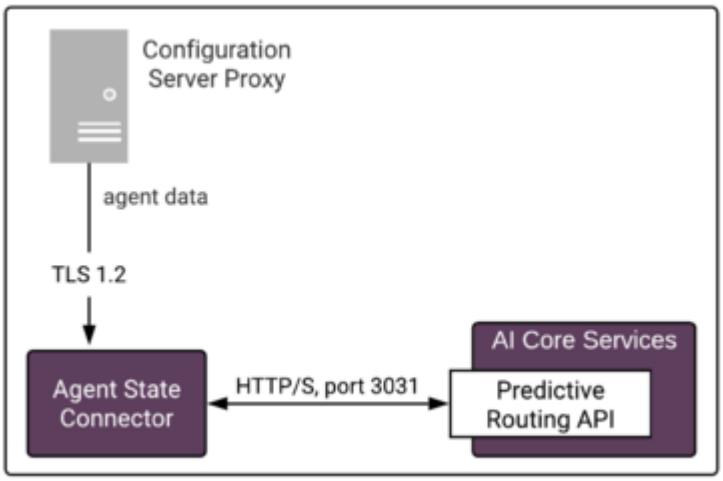
> **Important**
>
> Genesys recommends that you connect to Configuration Server Proxy, to reduce traffic on Configuration Server.

ASC is a Java application that can be monitored, started, and stopped in Solution Control Interface. It supports a warm-standby high availability architecture. Certain architectural details about ASC depend on the release you have deployed:

- In ASC 9.0.015.01 and lower, ASC connects to Stat Server to read agent availability data used in determining the preferred target agent and to configure and read the output for custom statistics.



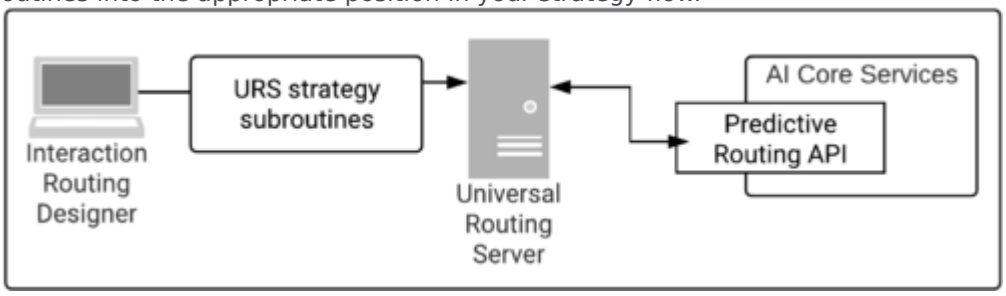
- In ASC 9.0.015.04 and higher, the connection to Stat Server is optional. If you do not add a Stat Server to the Connections tab of the ASC Application object, agent availability data is taken from Universal Routing Server (URS), reducing the number of connections required.



Subroutines Architecture

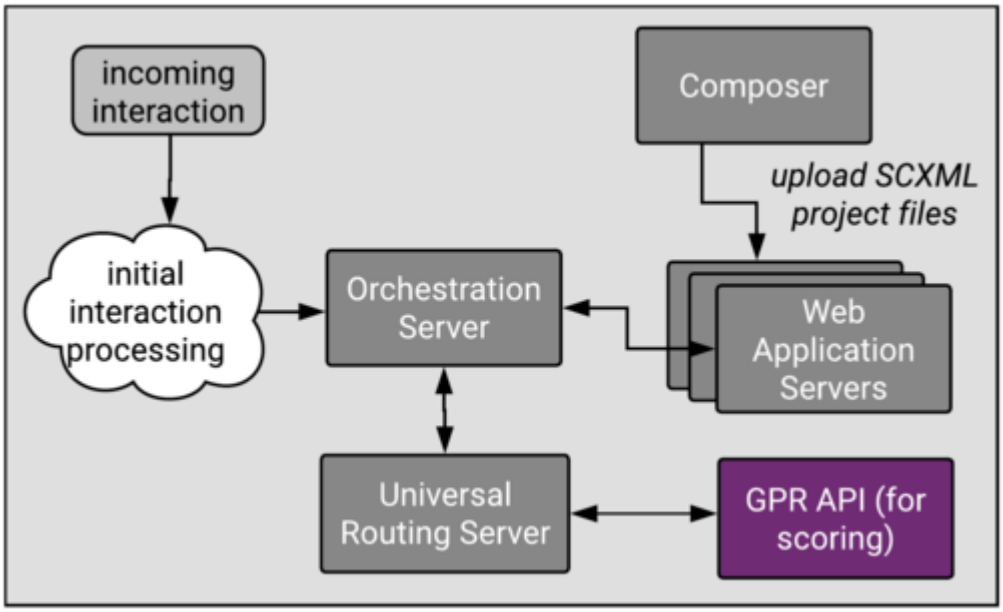
Predictive Routing supplies out-of-the-box subroutines for environments running either Interaction Routing Designer (IRD) + Universal Routing Server (URS) or Composer + Orchestration Server (ORS) + URS.

- IRD requires you to use the Predictive Routing URS Strategy Subroutines component. Insert the strategy subroutines into the appropriate position in your strategy flow.



- Composer requires the use of the Predictive Routing Composer Subroutines. Insert the subroutines into the appropriate position in your workflow. If you are using Composer, you need Orchestration Server (ORS) as well as URS in your environment.

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



The Subroutines invoke Predictive Routing in real time. They send a request to AICS, which performs the scoring based on the information you configured in your Predictor and the Model or Models based

on it. AICS 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

AI Core Services and Agent State Connector 9.0.015.05 and higher are delivered as a set of 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. To configure HTTPS connections, see Configuring GPR to Use HTTPS.
- GPR uses a CentOS 7 Docker image as the base image.
  - Genesys supports Security Enhanced Linux (SELinux) on CentOS 7. For a discussion of this functionality and how to configure it, see How to disable SELinux on the Linux web site.
- GPR Docker images containing Genesys software are continuously scanned for vulnerabilities as part of the build and test pipelines.
- All GPR Docker containers run in unprivileged mode.
- Inside Docker containers, GPR software is executed as a non-root user.
- All ports and volumes that should be exposed by each container are specified in Required Ports for Firewall Configuration.

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.

- GPR delivered as set of Docker containers does not require any additional ports to be open.
- GPR uses MongoDB as its database, which is also delivered as Docker image. GPR uses the official MongoDB Docker image at **https://hub.docker.com/_/mongo/**.
- MongoDB inside the Docker container requires access to the same ports and same hardware resources as MongoDB running outside of a Docker container.

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

To understand how MongoDB databases comply with various security regulations and best practices, see the following page on the MongoDB site:

- MongoDB Security

Secure Connections

Predictive Routing supports the following security and connection protocols:

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

The following protocols are supported for the specified connections:

- ASC to Config Server: TLS 1.2; you can specify an upgrade-mode Configuration Server port by updating the **-port** command line parameter in the ASC `Application` object **Start Info** tab.
- ASC to Stat Server: TLS 1.2
- ASC to AICS: HTTPS
- URS or ORS to AICS: HTTPS

Configure GPR to Use HTTPS

GPR supports HTTPS by default. The procedures linked below provide the required configuration to use HTTPS with GPR.

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

- Configure AICS to Use HTTPS
- Configure ASC to Use HTTPS
- Configure URS Strategy Subroutines/Composer Subroutines to Use HTTPS

Secure Logins

Predictive Routing supports LDAP authentication for user logins. See Settings: Configuring Accounts and Account: User Management for procedures to configure LDAP authenticated accounts.