Designer Deployment Guide

Deploy Designer (versions prior to v9010005)

6/14/2026

Contents

Learn how to deploy Designer as a service in a Kubernetes cluster (for **DesDepMnfst** versions prior to **v9010005**).

> **Important**
>
> For deployment instructions for **DesDepMnfst v9010005** and above, click here.

1. Prerequisites

Before deploying Designer, make sure the following resources are deployed, configured, and accessible:

1.1 Kubernetes cluster prerequisites

- Kubernetes 1.12+
- Helm 3.0
- Persistent volumes for workspace storage (minimum 2GB) and logs (minimum 5GB) configured in the cluster.
  - Each Designer and DAS pod will make persistent volume claims for storage and logs.
  - The volumes must be on shared storage (such as NFS) to enable changes made on one pod to become available on all other pods.
  - If a NFS server is used for shared storage, it should be deployed as highly available (HA) in order to avoid single points of failure.

> **Important**
>
> Genesys recommends using the ObjectiveFS (OFS) file system or any variant of the Network File System (NFS).

> **Important**
>
> The Designer manifest package includes sample YAML files to create an NFS server and persistent volumes.

1.2 Genesys components dependencies

- GWS 9.x
  - Configured to work with a compatible version of Configuration Server.
  - Contact Center provisioned in GWS (contact center ID available from GWS).
- ORS 8.1.400.x
- Nexus 9.x
- URS 8.1.400.x
- StatServer 8.5.11x.yz

1.3 External prerequisites

ElasticSearch 7.4.2 and 6.2.x for Designer Analytics and audit trails (optional and can be enabled later).

2. Deployment Process Overview

The Designer deployment process consists of the following steps:

1. Create roles for Designer.
2. Set up a transaction list.
3. Install Designer.
4. Install DAS.
5. Change the default values of the configurable parameters for Designer and DAS, if required.
6. Change the default values of additional configuration settings, if required.
7. Enable additional features.

Each of the above steps is explained in detail in the following sections.

3. Configuration Server objects

Designer uses roles and access groups to determine permissions associated with the logged-in user. To enable this, you must make these changes in GAX or CME.

3.1 Create roles for Designer

Designer support a number of bundled roles suitable for various levels of users.

- **DesignerDeveloper**Most users fall into this category. These users can create Designer applications, upload audio, and create business controls. They have full access to Designer Analytics.
- **DesignerBusinessUser**These users cannot create objects but they can manage them (for example, upload audio, change data tables, and view analytics).
- **DesignerAnalytics**These users only have access to Designer Analytics.
- **DesignerAdmin**These users can set up and manage partitions associated with users and Designer objects.
- **DesignerOperations**Users with this role have full access to all aspects of the Designer workspace. This includes the **Operations** menu (normally hidden), where they can perform advanced operational and maintenance tasks.

To create these roles, import the **.conf** files included in the **Designer Deployment Manifest** package. They are located in the **packages/roles/** folder.

In addition, ensure the following for user accounts that need access to Designer:

- The user must have read permissions on its own Person object.
- Users must be associated with one or more roles via access groups.
- The on-Premises user must have at least read access on the user, access group(s), and roles(s).

3.2 Set up a transaction list

Designer requires a transaction list for configuration purposes as described in other sections of this document. To set this up:

1. Create a transaction list called **DesignerEnv**.
2. Import the file **configuration/DesignerEnv.conf**, located in the Designer Deployment Manifest package.
3. Edit any values according to the descriptions provided in the **Additional configuration settings** section.
4. Save the list.
5. Ensure Designer users have at least read access to the **DesignerEnv** transaction list.

> **Important**
>
> The DesignerEnv transaction list is created under the *Transaction* root folder if the *Internal* folder does not exist.

4. Deploying Designer

This section describes how to deploy Designer on your Kubernetes cluster.

Ensure the following:

- Designer helm package is downloaded.
- Designer and DAS images are accessible from the cluster.

4.1 Install Designer and DAS

Install Designer using the following command (replace `designer-service` if you are using a different name for your Designer service):

```
helm install designer-service designer-9.0.11.xx.xx.tgz
```

Or

```
helm install designer-service -f designer-values.yaml designer-9.0.11.xx.xx.tgz.tgz
```

Next, install DAS using the following command (replace `das-service` if you are using a different name for your DAS service):

```
helm install das-service das-9.0.11.xx.xx.tgz
```

Or

```
helm install das-service -f das-values.yaml das-9.0.11.xx.xx.tgz
```

These commands deploy Designer on the Kubernetes cluster using the default configuration.

The Parameters section lists the parameters for both Designer and Designer Application Server (DAS) that can be configured during installation. It is recommended to add changed settings into a separate file (for example, **designer-values.yaml*)*** and specify that file while installing the chart.

4.1 Running Designer as a Non-Root User

You can run Desinger as a non-root user. Currently, only a **Genesys** user is supported by the Designer base image.

- By default Designer is run as a root user. To run it as a **Genesys** user, you must add the security context in the helm chart and configure the following in the **values.yaml** file:

```
runAsUser: 500
runAsGroup: 500
```

500 is the ID of the Genesys user and cannot be modified.

- The file system must reside within the Genesys user in order to run Designer as a Genesys user. Change the NFS server host path to the Genesys user:

```
chown -R genesys:genesys
```

- After installation, log in to the container and run `ps -ef` to verify if all processes are running as a Genesys user.

4.2 Running DAS as a Non-Root User

You can run DAS as a non-root user. Currently, only a **Genesys** user is supported by the Designer base image.

- To run DAS as a **Genesys** user, you must add the security context in the helm chart and configure the following in the **values.yaml** file:

```
runAsUser: 500
runAsGroup: 500
```

500 is the ID of the Genesys user and cannot be modified.

- After installation, log in to the container and run `ps -ef` to verify if all processes are running as a Genesys user.

5. Parameters

This section lists the configurable parameters of the Designer and Designer Application Server (DAS) chart and their default values.

Designer

Parameter	Description	Default
`deployment.replicaCount`	No. of services to be created	2
`deployment.strategy`	Rolling update / re-create	RollingUpdate
`desImage.repository`	Docker repository for Designer	pureengage-docker-staging.jfrog.io/designer/designer
`desImage.tag`	Designer Image version	9.0.109.08.20
`volumes.workapceMountPath`	Designer workspace path inside the container	/designer/workspace
`volumes.workspaceClaim`	Persistent volume claim name for the workspace	designer-managed-disk
`volumes.logMountPath`	Designer log path inside the container	/designer/logs
`volumes.logClaim`	Persistent volume claim name for logs	designer-logs
`healthApi.path`	Health check request to be sent	/health
`healthApi.containerPort`	Container running port	8888

`healthApi.startupDelay`	Health will be started after a given delay	20
`healthApi.checkInterval`	The interval between each health check requests	5
`healthApi.failureCount`	No of health check failure to mark the container as instable or restart	5
`designerEnv.enabled`	Enables the ConfigMap based env input	true
`designerEnv.configName`	Name of the ConfigMap	designer-config
`designerEnv.envs.DES_PORT`	Designer port for container	8888
`designerEnv.envs.DES_APPSERVER_DAS_HOST`	DAS hostname	das
`designerEnv.envs.DES_APPSERVER_DAS_PORT`	DAS port	80
`designerEnv.envs.DES_USE_HTCC`	To enable GWS based auth	true
`designerEnv.envs.DES_HTCC_SERVER`	GWS server URL	gws-service-proxy
`designerEnv.envs.DES_HTCC_PORT`	GWS server port	80
`designerEnv.DES_GWS_CLIENT_ID`	GWS Client ID Create a new client ID if the default does not work. Follow the steps in the link below, to create new GWS client credentials: Creating Client for Provisioning Service	external_api_client
`designerEnv.DES_GWS_CLIENT_SECRET`	GWS Client secret	****
`service.type`	Service port either CluserIP/ NodePort/LoadBalancer	NodePort
`service.port`	Designer service to be exposed	8888
`service.targetPort`	Designer application port running inside the container	http
`service.nodePort`	Port to be exposed in case service.type=NodePort	30180
`ingress.enabled`	Enable/Disable ingress	true
`ingress.paths`	Ingress path	/
`ingress.hosts`	Hostname	ssdev1.genhtcc.com
`ingress.tls`	TLS based security enabling	nil
`resources.limits.cpu`	Maximum amount of CPU K8s allocates for container	600m
`resources.limits.memory`	Maximum amount of Memory K8s allocates for container	1Gi
`resources.requests.cpu`	Guaranteed CPU allocation for container	400m
`resources.requests.memory`	Guaranteed Memory allocation for container	512Mi
`nodeSelector`	To allow Pods to be scheduled on the nodes based labels assigned	Default value:

	to nodes.	`nodeSelector: {}` Sample value: `nodeSelector: :`

Designer Application Server (DAS)

Parameter	Description	Default	
`deployment.replicaCount`	No of service to be created	2	
`dasImage.repository`	Docker repository for DAS	pureengage-docker-staging.jfrog.io/designer/das	
`dasImage.tag`	DAS Image version	9.0.106.03.7	
`dasVolumes.workapceMountPath`	DAS workspace path inside the container	/das/www/workspaces	
`dasVolumes.workspaceClaim`	Persistent volume claim name for the workspace	designer-managed-disk	
`dasVolumes.logMountPath`	DAS log path inside the container	/das/log	
`dasVolumes.logClaim`	Persistent volume claim name for logs	designer-logs	
`dasHealthApi.path`	Health check request to be sent	/health	
`dasHealthApi.containerPort`	Container running port	80	
`dasHealthApi.startupDelay`	Health will be started after a given delay	20	
`dasHealthApi.checkInterval`	The interval between each health check requests	5	
`dasHealthApi.failureCount`	No of health check failure to mark the container as instable or restart	5	
`dasService.type`	Service port either CluserIP/NodePort/ LoadBalancer	NodePort	
`dasService.port`	DAS service to be exposed	80	
`dasresources.limits.cpu`	Maximum amount of CPU K8s allocates for container	600m	
`dasresources.limits.memory`	Maximum amount of Memory K8s allocates for container	1Gi	
`dasresources.requests.cpu`	Guaranteed CPU allocation for container	400m	
`dasresources.requests.memory`	Guaranteed Memory	512Mi	

	allocation for container		

`nodeSelector`	To allow Pods to be scheduled on the nodes based labels assigned to nodes.	Default value:
		`nodeSelector: {}`
		Sample value:
		`nodeSelector:`
		`:`	

6. Additional configuration settings

Post deployment, Designer configuration is managed in two locations:

- /designer/flowsettings.json
- Configuration Server in the Tenant/Transactions/Internal/DesignerEnv transaction list

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
Analytics	enableAnalytics	Yes			Flag to enable analytics.
Analytics	esUrl	Yes			Elasticsearch URL (for example, `http://es-service:9200`).
Analytics	esServer	Yes			Elasticsearch Server HostName (for example, `es-service`).
Analytics	esPort	Yes			Elasticsearch port (for example, `9200`).
Analytics	ReportingURL		Yes	reporting	URL of Elasticsearch where Designer applications will report data (for example, http://es-service:9200).
Analytics	esMaxQueryDuration				The maximum time range (in days) to query in Designer Analytics. Data for each day is stored in a

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
					separate index in Elasticsearch.
Analytics	sdrMaxObjCount				The maximum count of nested type objects to be captured in SDRs.
Analytics	SdrTraceLevel				This caps the level of detail captured in analytics.
Audio	useUserRecordedSystemAudio				
Audit	enableESAuditLogs	Yes			Enable or Disable Audit logs captured in Elasticsearch.
Audit	enableFSAuditLogs	Yes			Enable or Disable Audit logs captured in the file system.
Audit	maxAppSizeCompare				The maximum size of a data object for which a differential will be captured in audit logs.
Audit	enableReadAuditLogs	Yes			Control whether reading of objects is captured in audit trails.
Authorization	disableRBAC	Yes			Controls if Designer reads and enforces permissions associated with the logged in user's roles.
Authorization	disablePBAC	Yes			Controls if Designer allows partitioning of Designer workspace and restricts a

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
					user's access to Designer objects to the user's partitions.
Collboration	locking	Yes			The type of locking used, for an editing session of applications, modules, or data tables.
DAS	applicationHost	Yes			The server name Designer uses to generate the URL to the application. ORS and MCP fetch application code and other resources from this URL.
DAS	applicationPort	Yes			The corresponding port to be used with applicationHost.
DAS	deployURL	Yes			This is normally not changed. It is the relative path to the workspace on DAS.
Deployment	ssoLoginUrl	Yes			URL of GWS authentication UI. Designer redirects to this URL for authentication.
Digital	rootsSRL	Yes			If specified, this is used to filter which Root Categories to display when selecting Standard Responses.

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
Digital	maxFlowEntryCount		Yes	flowsettings	Specifies how many times the same application can process a specific digital interaction.
External APIs	httpProxy	Yes	Yes	flowsettings	Specifies the proxy used for external request and nexus API calls (if enable_proxy is `true`).
External APIs	redundantHttpProxy	Yes	Yes	flowsettings	Specifies the backup proxy used for external request and Nexus API calls (if enable_proxy is `true`), when httpProxy is down.
Features	features				This is an object. See the Features section for a list of supported features.
GWS	usehtcc	Yes			Set to `true` so Designer works with GWS. If set to `false`, Designer defaults to a local mode and may be used temporarily if GWS is unavailable.
GWS	htccServer	Yes			GWS Server
GWS	htccport	Yes			GWS Port
GWS	maxConcurrentHTCCRequest	Yes			For batch operations to GWS, the maximum number of

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
					concurrent requests that Designer will send to GWS.
GWS	batchOperationResultTTL	Yes			For batch operations to GWS, the time (in milliseconds) that Designer stores results of a batch operation on the server, before deleting it.
Help	docsMicroserviceURL	Yes			URL for Designer Documentation
IVR	recordingType				Specifies the recording type to be used in the Record block. Set as GIR. If the option is missing or blank, Full Call Recording type is used.
Nexus	url		Yes	nexus	URL of Nexus that typically includes the API version path (e.g. https://nexus-server/nexus/api/v3).
Nexus	password		Yes	nexus	Nexus x-api-key created by Nexus deployment.
Nexus	enable_proxy		Yes	nexus	Boolean value on whether httpProxy is used to reach Nexus.
Nexus	profile		Yes	nexus	Enable Contact Identification via Nexus (e.g. to enable Last

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
					Called Agent routing).
Process	port	Yes			Designer process port in the container. Typically, you should keep the default value.
Provisioning	primarySwitch				Specify the primary switch name if more than one switch is defined for the tenant. Designer fetches and works with route points from this switch.
Routing	ewtRefreshTimeout		Yes	flowsettings	Specifies the interval (in seconds) to refresh the Estimated Waiting Time when routing an interaction.
Security	zipFileSizeLimitInMegaBytes				Defines the maximum zipFile size limit (in megabytes) during bulk audio import.
Security	tempUploadDir				The path where the zipFile is stored during bulk audio import process.
Security	disableCSRF	Yes			
Security	disableSecureCookies	Yes			Disable the secure cookies header.
Session	idleTimeout	Yes			Idle timeout (in seconds) before a user session is terminated

Category	Setting Name	flowsettings.json	DesignerEnv	DesignerEnv Section	Description
					while editing applications, modules, or data tables.
Session	lockTimeout	Yes			Timeout (in seconds) before a resource lock is released, for an editing session of applications, modules, or data tables.
Session	lockKeepalive	Yes			Interval (in seconds) before the client sends a ping to the server, to refresh the lock for an editing session of applications, modules, or data tables.
Tenancy	multitenancy	Yes			Should be set to `true`.
Tenancy	localmode	Yes			Should be set to `false`.
Workflow	maxBuilds	Yes			Specifies the maximum number of builds permitted per application.
Workflow	enablePTE		Yes	flowsettings	Boolean value on whether PTE objects are enabled at runtime.

7. Features

The features specified here must be configured under the `features` object in the `flowsettings.json` file.

For example,

"features": {

"callbackv2": true,

..

..

}

> **Important**
> These features are configured only in the flowsettings.json file and not in DesignerEnv.

Category	Feature Setting Name	Description	Default Value
Audio	enableBulkAudioImport	Enable or disable the bulk audio import feature.	false
Audio	grammarValidation	If enabled, Designer will validate invalid grammar files during grammar upload. If it is enabled, only valid grammar files (.grxml or Nuance compiled binary grammar) can be uploaded.	false
Nexus	nexus	Enable or disable Nexus.	false

The Personas feature is enabled during a new tenant creation in Azure. The following are performed during workspace initialization:

- The Personas feature flag is enabled in `tenantsettings.json`.
- The GTTS only *personas.json* file is copied to `workspace/tenant ccid/workspace/personas/personas.json`.
- The defaultPersona setting is configured in the DesignerEnv transaction list (`flowsettings->defaultPersona = Gabriela`).

8. Upgrades

To upgrade the service when a new Designer/DAS Helm chart is released:

`helm upgrade`

To upgrade when a new Designer/DAS image version is released:

Option 1- Using the default settings (recommended)

1. Modify the image tag parameter in the **designer-values.yaml** file.
   For example, if you are upgrading the Designer version, modify `tag` under the `desImage` section. For upgrading DAS, modify the `tag` under `dasImage` section.
2. `helm upgrade -f designer-values.yaml`

Option 2 - Using the SET command for Helm

For Designer,

```
helm upgrade designer-service designer-9.0.11.xx.xx.tgz --set desImage.tag=
```

For DAS,

```
helm upgrade das-service das-9.0.11.xx.xx.tgz --set dasImage.tag=
```

9. Uninstall

To uninstall a service:

`helm uninstall`