CX Contact Deployment Guide

Deploying with Docker Compose

4/13/2025

Contents

Deploying with Docker Compose

To deploy CX Contact by using Docker Compose, complete the following deployment procedures. The first three procedures are common to both deployment methods. Click the link for the appropriate topic:

Summary of deployment procedures
1. Ensure the Prerequisites are met
2. Create the Outbound Database
3. Create the Outbound Database Access Point
4. Start Outbound Contact Server (OCS)
5. Deploy with Docker Compose

Deploy with Docker Compose

Summary of procedures: Deploy with Docker Compose
1. Obtain the Docker Compose scripts • Use common CX Contact commands • Obtain images in disconnected environments
2. Set up the environment: • Use automatic setup (Genesys strongly recommends that you use this method to set up the environment.) • Use manual setup
3. Log in to CX Contact

Important

Contact Genesys Customer Care regarding downloading CX Contact and GWS Docker images. These images should be pushed to the local registry. Refer to https://docs.docker.com/registry/deploying/ for details. Make note of the GWS Components versions. You may need to enter these versions when performing the initial setup.

Docker Compose scripts

To receive all of the latest files required for the Docker Compose deployment, you must first copy the **Docker Compose** scripts.

To obtain the Docker Compose scripts:

1. Execute the following Bash commands in the order that they appear here:
   - `$ export DEPLOY_CXCONTACT_IMAGE=<deploy_cx_contact_image>`
   - `$ echo "docker run --rm -it -u $(id -u ${USER}):$(id -g ${USER}) -e init=true -v "$(pwd)":/env:rw $DEPLOY_CXCONTACT_IMAGE" > cxc-app.sh`
   - `$ bash cxc-app.sh`

   When asked if you would like to replace with updates found for the **cxc-app.sh** script select **2** to replace.
   - The old file is moved to the `/backup/MMDDYY-hhmm` folder.
     Note: Each update is saved in a new directory. The name of the new directory is the date on which the update was performed and completed.

2. Once you obtain the Docker Compose script make it executable by running the following command:

   ```
   $ chmod +x cxc-app.sh
   ```

   Then, execute the following **Help** command to obtain all of the available commands:

   ```
   $ ./cxc-app.sh help
   ```

   Help Output

   ```
   Welcome to CXContact deployment service. Following commands are available:

   init <deploy_cx_contact image>            - Initial setup
   start                                     - Start CXContact docker-compose environment
   stop                                      - Stop CXContact docker-compose environment
   restart                                   - Restart CXContact docker-compose environment
   status                                    - Get status of all running containers
   provision <deploy_cx_contact image>       - Provisioning CXContact
   cxc-only [on/off]                         - Switch between cxc-only deployment and
   single node (with GWS services) deployment

   save <optional parameters>                - Save docker images in archive
   Available optional parameters for save:
   --only <tag> for e.g cxcontact.            - Will save only specific images
   -o,--output <name>                         - Output names for archive with images and
   import script
   -t,--tag <tag>                             - Will apply new tag to images, for internal
   registries

   Note! <deploy_cx_contact image> parameter is optional. Default - will be used latest
   local image
   If no image found - you will be asked to prompt image name to pull.
   ```

3. Execute the following command to obtain the CX Contact Docker Compose **.yaml** files:

   ```
   $ ./cxc-app.sh init
   ```

At this point, the following occurs:

- You will be prompted for a CX Contact Deployment Service image (if it is not present locally).

Note: The Docker Compose script uses the latest `deploy_cx_contact` image. To use a specific image run `./cxc-app.sh init <image_name>`.

- The Docker Compose script pulls the Deployment Service image and verifies which deployment should be initialized (that is, Docker or Kubernetes).
- The **.yaml** files are copied to the Docker or Kubernetes folder in the same directory in which the Docker Compose script is located.
- You ae asked to configure CX Contact. The default values are suggested for each environment variable. You can replace the default values with values applicable for your environment.

Note: If you are using the local Docker registry, specify it as the value of CXC_DOCKER_REPOSITORY and GWS_DOCKER_REPOSITORY variables. Change default values of GWS components versions to the actual versions of GWS Components images pushed to the local registry.

4. Start the CX Contact Docker Compose Environment by running the following command:

```
$ ./cxc-app.sh start
```

5. By default, as of CX Contact 9.0.025, the Docker Compose **.env** file is configured for CDP NG connectivity. The following default values show how CDP NG is used.

```
# COMPLIANCE DATA SETTINGS
 # List Builder embedded CDP_NG Compliance Data
 EMBEDDED_COMPLIANCE_DATA_BASEPATH="/list_builder/data/ng_init_data"
 # OPTIONAL: List Builder embedded LEGACY CDP Compliance Data
 #EMBEDDED_COMPLIANCE_DATA_BASEPATH="/list_builder/init_data"
 # Compliance data rule sets:
 AREACODE_RULE_SET="AU,CA,GB,NZ,US"
 GEO_RULE_SET="AU,CA,GB,NZ,US"
 POSTAL_RULE_SET="CA,GB,US"
 DNC_RULE_SET="GB,US"
 # In order to switch to Legacy CPD, change CDP_NG_URL and CDP_NG_GCLOUD_AUTH to empty
 values
 CDP_NG_URL="https://api.usw2.pure.cloud/api/v2/outbound/compliancedata"
 CDP_NG_GCLOUD_AUTH="https://login.usw2.pure.cloud/oauth/token"
```

- CDP_NG_GCLOUD_ID and CDP_NG_GCLOUD_SECRET are required parameters and do not have a default value.
- These parameters must be requested by creating a CLOUDCON ticket before attempting to upgrade to CX Contact 9.0.025+ or before deploying CX Contact 9.0.025+ for the first time.
- The new CLOUDCON ticket must include the customer name and the name of the person who will receive the keys in the Jira ticket. CDP_NG_GCLOUD_ID=<Must be provided> CDP_NG_GCLOUD_SECRET=<Must be provided>
- To return to CDP Legacy, use the following configuration parameters:

```
EMBEDDED_COMPLIANCE_DATA_BASEPATH="/list_builder/init_data"
 CDP_NG_URL=
 CDP_NG_GCLOUD_AUTH=
 CDP_NG_GCLOUD_ID=
 CDP_NG_GCLOUD_SECRET=
```

Common CX Contact commands

CX Contact Procedure	Command
Start CX Contact	$ ./cxc-app.sh start
Stop CX Contact	$ ./cxc-app.sh stop
Restart CX Contact	$ ./cxc-app.sh restart
View current CX Contact status and uptime.	$ ./cxc-app.sh status
Switch between single host deployment and multi host deployment (that is, when GWS and CX Contact are deployed on different hosts).	$ ./cxc-app.sh cxc-only [on/off]
Provision CX Contact	$ ./cxc-app.sh provison **Note:** Provisioning can be executed multiple times.
Revalidate the configuration when provisioning fails.	$ ./cxc-app.sh init
Pull images in a disconnected environment.	$ ./cxc-app.sh init $ ./cxc-app.sh save <optional parameters> **Note:** The images are archived and an be transferred to a Docker host that does not have an internet connection.

Obtain Images in Disconnected Environments

Deployments that are not connected to the Internet must obtain images from the Genesys Engage Docker Repository and archive them. The archived images are then transferred to a Docker host that is not connected to the Internet.

To obtain images from the Genesys Engage Docker Repository run the following command on a computer with access to the Internet and save the images in an archive location.

```
$ ./cxc-app.sh init
$ ./cxc-app.sh save <optional>
```

The archived images must then be transferred to a destination host and used for deployment.

> ### Important
> To store the images in your own Docker registry (only required for Kubernetes deployments or your own Docker registry), you can add the **--tag** parameter with your docker-registry. The archived images must be transferred to a destination host and used for deployment.

Set up the environment

Set up the environment either automatically (recommended) or manually. Procedures for each option are provided below.

Set up the environment automatically (Recommended)

If you have already deployed CX Contact using Docker Compose, start at step 1 below. If not, see Docker Compose scripts for details.

1. Obtain the CX Contact Docker Compose **.yaml** files by executing the following command:

   ```
   $ ./cxc-app.sh init
   ```

2. While executing **$ ./cxc-app init.sh**, select the Docker deployment and select **y** when asked if you would like to configure CX Contact.
3. In the directory containing **cxc-app.sh**, execute the following:

   ```
   ./cxc-app.sh provision
   ```

Provisioning will take approximately 1 to 2 minutes.

Set up the environment manually

> **Important**
>
> Genesys recommends this option for advanced users only.

To set up the environment manually, you'll make API requests to GWS. Requests should point to the host name or IP address of the external load balancer for GWS or CX Contact. In Docker Compose deployments, this is the VM where Docker runs.

Complete the procedures in the summary below.

Summary of procedures: Set up environment manually
1. Verify successful start of gws-postgres
2. Check gws-core-environment
3. Check gws-core-auth
4. Create the environment
5. Create the contact center
6. Create the authentication client
7. Create the external_api_client
8. Get the access token for cx_contact

Summary of procedures: Set up environment manually
9. Verify authentication (Optional)
10. Verify clients
11. Create the tenant in api-aggregator
12. Configure session profile
13. Add location in CloudCluser Application

Verify successful start of gws-postgres

To verify the successful start of **gws-postgres**:

```
docker-compose logs gws-postgres
```

This is the expected response:

```
gws-postgres_1                | LOG:  database system is ready to accept connections
gws-postgres_1                | LOG:  autovacuum launcher started
```

Check gws-core-environment

To check **gws-core-environment**:

```
curl http://localhost:8091/environment/v3/version
```

Check gws-core-auth

To check **gws-core-auth**:

```
curl  'http://localhost:8095/auth/v3/oauth/
authorize?response_type=code&client_id=cx_contact&redirect_uri=http://localhost/api-
aggregator/v2/login-callback'
```

Check that there is a reply. It will be an error, but it must be received.

Create the environment in GWS

Send a POST request to GWS.

Pass a body parameter in JSON format, called **data**, with the following properties:

Property	Value
username	The super administrator account name for Configuration Server.
appName	Cloud
password	Use the password for the super administrator

Property	Value
	account name.
connectionProtocol	addp
localTimeout	7
remoteTimeout	11
traceMode	CFGTMBoth
tlsEnabled	false
primaryPort	Configuration Server's TCP Listener port.
readOnly	false
primaryAddress	Configuration Server's host name or IP address
locations	/USW1
tenant	Environment

Code sample

```
curl  --user ops:ops  -H "Content-Type: application/json" -X POST http://localhost:8091/
environment/v3/environments  -d  '
{
  "data": {
    "username": "default",
    "appName": "Cloud",
    "password": "password",
    "connectionProtocol": "addp",
    "localTimeout": 7,
    "remoteTimeout": 11,
    "traceMode": "CFGTMBoth",
    "tlsEnabled": false,
    "configServers": [
      {
        "primaryPort": 8888,
        "readOnly": false,
        "primaryAddress": "10.51.30.154",
        "locations": "/USW1"
      }
    ],
    "tenant": "Environment"
  }
}
'
```

Expected response

```
{"status":{"code":0},"path":"/environments/bf032640-9073-435d-9447-718b7cc7dc43"}
```

> **Important**
>
> Take note of the environment ID parameter in the response (**bf032640-9073-435d-9447-718b7cc7dc43**) – it is required for subsequent requests.

Create the contact center

Send a POST request to GWS.

Pass a body parameter in JSON format, called **data**, with the following properties:

Property	Value
environmentid	bf032640-9073-435d-9447-718b7cc7dc43
auth	configServer

Code sample

```
curl  --user ops:ops  -H "Content-Type: application/json" -X POST http://localhost:8091/
environment/v3/contact-centers  -d  '
{
  "data": {
    "environmentId": "bf032640-9073-435d-9447-718b7cc7dc43",
    "domains": ["domain.com"],
    "auth": "configServer"
  }
}
'
```

Expected response

This produces the following result:

```
{"status":{"code":0},"path":"/contact-centers/3952ccd2-a34a-46c1-b51e-8917628554c9"}
```

Create the authentication client

To create the authentication client, send a POST request to GWS.

Pass a body parameter in JSON format, called **data**, with the following properties:

Property	Value
clientType	*
internalClient	true
authorizedGrantTypes	refresh_token, implicit, password, client_credentials, authorization_code
redirectURIs	http://10.11.64.16
authorities	ROLE_INTERNAL_CLIENT
description	cx_contact
accessTokenExpirationTimeout	43200
refreshTokenExpirationTimeout	2592000
name	cx_contact
client_id	cx_contact
client_secret	<client secret token>

Code sample

```
curl --user ops:ops -X POST   http://localhost:8095/auth/v3/ops/clients/   -H 'Cache-Control:
no-cache'   -H 'Content-Type: application/json'    -d '
{"data": {
        "clientType": "CONFIDENTIAL",
        "scope": [
            "*"
        ],
        "internalClient": true,
        "authorizedGrantTypes": [
            "refresh_token",
            "implicit",
            "password",
            "client_credentials",
            "authorization_code"
        ],
      "redirectURIs": [
        "http://10.11.64.16"
      ],
        "authorities": [
            "ROLE_INTERNAL_CLIENT"
        ],
        "description": "cx_contact",
        "accessTokenExpirationTimeout": 43200,
        "refreshTokenExpirationTimeout": 2592000,
        "name": "cx_contact",
        "client_id": "cx_contact",
        "client_secret" : "<client secret token>"
    }
}'
```

Expected response

The expected response is **200 OK**.

Create external_api_client

To create the **external_api_client** for communication with GWS services, send a POST request.

Pass a body parameter in JSON format, called data, with the following properties:

Property	Value
clientType	CONFIDENTIAL
scope	*
internalClient	true
authoraizationGrantTypes	refresh_token, implicit, client_credentials, password, authorization_code
authorities	ROLE_INTERNAL_CLIENT
description	external_api_client
accessTokenExpirationTimeout	43200
refreshTokenExpirationTimeout	2592000

Property	Value
name	external_api_client"
client_id	external_api_client"
client_secret	client secret token

Code sample

```
curl   --user ops:ops  -H "Content-Type: application/json"  -X POST http://localhost:8095/
auth/v3/ops/clients  -d '
{
      "data":
      {
         "clientType": "CONFIDENTIAL",
          "scope": ["*"],
          "internalClient": true,
          "authorizedGrantTypes":  [
                  "refresh_token", "implicit", "client_credentials",
                   "password", "authorization_code"
          ],
          "authorities": ["ROLE_INTERNAL_CLIENT"],
          "description": "external_api_client",
          "accessTokenExpirationTimeout": 43200,
          "refreshTokenExpirationTimeout": 2592000,
          "name": "external_api_client",
          "client_id": "external_api_client",
          "client_secret": "client secret token"
       }
}
'
```

Expected response

The expected response is **200 OK**.

Get access token for cx_contact

To get the access token for **cx_contact**, send a POST request to GWS:

```
curl --user cx_contact:<client secret token>   -H "Content-Type: application/json"  -X POST
'http://localhost:8095/auth/v3/oauth/
token?grant_type=client_credentials>ope=*&client_id=cx_contact&client_secret=<client secret
token>'
```

This is the response:

```
{"access_token":"<bearer token>","token_type":"bearer","expires_in":43199,"scope":"*"}
```

Verify authentication (optional step)

To verify that authentication was successful, send a POST request to GWS:

```
curl --user external_api_client:secret  -H "Content-Type: application/json"  -X POST
'http://localhost:8095/auth/v3/oauth/
```

```
token?grant_type=client_credentials>ope=*&client_id=external_api_client&client_secret=<client
secret token>'
```

Verify clients

To verify that clients were created successfull, send a POST request to GWS

```
curl --user ops:ops http://localhost:8095/auth/v3/ops/clients   |   python -m json.tool
```

Create tenant in api-aggregator

To create the tenant in **api-aggregator**, send a POST request.

Pass a body parameter in JSON format, called data, with the following properties:

Parameter	Value
envrionmentid	bf032640-9073-435d-9447-718b7cc7dc43
shortTenantName	The short tenant name (for example 22-06).
customerName	The short tenant name (for example 22-06).
contactCenterId	The unique ID generated when a request is sent to GWS to create an Environment/Contact Center (for example 3952ccd2-a34a-46c1-b51e-8917628554c9).

Sample code

```
curl -X POST  -H "Authorization: Bearer  <bearer token>"  -H "Content-Type: application/
json"  -H 'Cache-Control: no-cache'  http://localhost:8102/api-aggregator/v2/tenants   -d '
{
   "data":
   {
        "domains": ["domain.com"],
        "environmentId": "bf032640-9073-435d-9447-718b7cc7dc43",
        "shortTenantName": "22-06",
        "customerName": "tenant_22-06",
        "contactCenterId": "3952ccd2-a34a-46c1-b51e-8917628554c9"
    }
}
'
```

Expected response

The expected response is **200 OK**.

Configure session profile

During CX Contact provisioning a set of objects is created in the Configuration Server.

Configure the Annex of the **DefaultSessionProfile** object (type=script), by replacing all **-1** with the following correct DBIDs:

- Voice Transfer Destination DN (origDNDBID)
- Trunk Group DBID (trunkGroupDNDBID)
- Statistics Server DBID (statServerDBID)
- Remove -1 from "serverDBIDs"

For example:

```
"data": {
  "interactionQueueDBID": 0,
  "origDNDBID": -1,
  "trunkGroupDNDBID": -1,
  "operationMode": 1,
  "statServerDBID": -1,
  "serverDBIDs": [
    -1
  ],
  "IVRProfileDBID": 0,
  "numOfChannels": 10
},
"isDefault": true
```

Add location in the CloudCluster Application

1. In the CloudCluster application, open the properties of the Connection to the OCS application.
2. In **Advanced -> Application Parameters**, enter the location using its short region name—for example: `locations=/USW1`

Log in to CX Contact

Log in to the CX Contact user interface with the URL `http://<your-docker-hostname>/ui/cxcontact/`

> **Important**
> You must include the backslash (/) after **cxcontact** (cxcontact/)