Sitecore Connect for Microsoft Dynamics 365 for Sales 7.0 Container Deployment Guide

A guide to deploying Sitecore Connect for Microsoft Dynamics 365 for Sales to Docker and Azure Kubernetes Service

November 29, 2021

Table of Contents

1. Introduction

Sitecore Connect for Microsoft Dynamics 365 for Sales (DCRM) enables you to synchronize data between Microsoft Dynamics systems and Sitecore.

This guide shows you how to add the DCRM connector to Sitecore container installations for Docker and Azure Kubernetes Service.

2. Prepare to deploy DCRM to Sitecore containers

This section explains what you need to deploy the Sitecore Connect for Microsoft Dynamics 365 for Sales (DCRM) connector to Sitecore containers for Docker and Azure Kubernetes Service.

2.1. Requirements

Before you add the DCRM module for Docker or AKS, you must have the following:

- Docker Desktop installed and running. For instructions on how to set up the Docker environment, see the Containers in Sitecore development documentation.
- if the installation is done on Docker, you must have the Sitecore Docker container files deployed on a local machine . For instructions on how to prepare the Sitecore containers, see the *Installation Guide for Developer Workstation with Containers* on the Sitecore download pag.
- If the installation is done on Kubernetes, you must have the Sitecore AKS container files deployed on a local machine . For instructions on how to prepare a Sitecore environment with Kubernetes, see the *Installation Guide for Production Environment with Kubernetes* on the Sitecore download page.

3. Add the DCRM connector module to Sitecore in Docker

To add Sitecore Connect for Microsoft Dynamics 365 for Sales (DCRM) in Docker, you must do the following in this order:

- Prepare the installation files.
- Build the Docker images.
- Update the Solr indexes.

3.1. Prepare the installation files

To prepare the files you need for the installation:

1. Download the DCRM container deployment package from the Sitecore download page. Extract it to your local workstation with the folder structure intact.
2. Go to the folder that you extracted the DCRM container deployment package to. Go to the folder for the Windows version and topology you are using, for example, `dcrm\compose\ltsc2019\xp1`.
3. Open the `.env-example` file in an editor. Copy all the variables to the clipboard.
4. Go to the Sitecore Experience Platform (SXP) container deployment folder on your local machine. Go to the folder for the Windows version and topology you are using, for example, `compose\ltsc2019\xp1`.
5. Open the `.env` file in an editor, and paste in the variables from the DCRM `.env-example` file.
6. Fill in the parameters for the following variables:
   - `ConnectionString_DCRM`
   - `ConnectionString_StagingDB`
7. Add the following variables to the `.env` file:

```
DEF_IMAGE=scr.sitecore.com/sxp/modules/sitecore-def-xp1-assets:7.0.0-1497
DCRMCNN_IMAGE=scr.sitecore.com/sxp/modules/sitecore-dcrm-xp1-assets:7.0.0-1497
TOOLING_IMAGE=scr.sitecore.com/tools/sitecore-docker-tools-assets:10.2.0-1809
```

8. Save the `.env` file.
9. From the DCRM `compose\<version>\<topology>` folder, copy the `docker-compose.override.yml` file to the SXP container deployment `compose\<version>\<topology>` folder (where the `docker-compose.yml` file is).

3.2. Build the Docker images

When you have prepared the installation files, you must create Docker files for each role, and build the Docker images.

> **NOTE**
> For more information on image assets, see the documentation on how to Add Sitecore Modules.

To build the images:

1. Go to the SXP container deployment folder on your local machine. Go to the folder for the Windows version and topology you are using, for example, `compose/ltsc2019/xp1`. Create a subfolder and name it `module`.
2. In the `module` folder, create these subfolders:
   - `cm`
   - `xconnect`
   - `xdbsearchworker`
   - `mssql-init`
3. In each subfolder, create a new file and name it `Dockerfile`.
4. In the `cm` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG DCRMCNN_IMAGE
ARG DEF_IMAGE
ARG TOOLING_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${DCRMCNN_IMAGE} as dcrm
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE} as baseImage

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]

WORKDIR C:\inetpub\wwwroot

# Add DEF module
COPY --from=def \module\cm\content

# Add DCRM module
COPY --from=dcrm  \module\cm\content

#Copy transformation files
COPY --from=dcrm  \module\xdttransform\cm\transforms\ C:\transforms\

#Add tools from sitecore-docker-tools-assets
COPY --from=tooling \tools\ C:\tools\

RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\inetpub\wwwroot -XdtPath \transforms
\cm
```

5. In the `xconnect` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG DCRMCNN_IMAGE
ARG TOOLING_IMAGE

FROM ${DCRMCNN_IMAGE} as dcrm
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE} as baseImage

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]

#Add tools from sitecore-docker-tools-assets
COPY --from=tooling \tools\ C:\tools\

# Copy models file into index worker
COPY --from=dcrm  \module\models C:\inetpub\wwwroot\App_Data\Models\

#Copy transformation files
COPY --from=dcrm  \module\transforms\ C:\transforms\

# Update value for IndexPIISensitive and IndexAnonymousContactData in IndexerSetting
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\inetpub\wwwroot -XdtPath
C:\transforms\xconnect
```

6. In the `xdbsearchworker` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG DCRMCNN_IMAGE
ARG TOOLING_IMAGE

FROM ${DCRMCNN_IMAGE} as dcrm
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE} as baseImage

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]

#Add tools from sitecore-docker-tools-assets
COPY --from=tooling \tools\ C:\tools\

#Copy transformation files
COPY --from=dcrm  \module\transforms\ C:\transforms\

# Update value for IndexPIISensitive and IndexAnonymousContactData in IndexerSetting
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\service -XdtPath C:\transforms
\xdbsearchworker
```

7. In the `mssql-init` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`
ARG BASE_IMAGE
ARG DCRMCNN_IMAGE

FROM ${DCRMCNN _IMAGE} as dcrm
FROM ${BASE_IMAGE}

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
```

```
# Deploy DCRM Staging database file
COPY --from=dcrm  C:\module\db C:\resources\dcrm
```

8. In the `compose\<version>\<topology>\docker-compose.override.yml` file, add build instructions for each role. For all topologies, start with these instructions:

```
version: "2.4"
services:
  cm:
    image: sitecore-dcrm-xp1-cm:${SITECORE_VERSION}
    build:
      context: ./module/cm
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-cm:${SITECORE_VERSION}
        DEF_IMAGE: ${DEF_IMAGE}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
        TOOLING_IMAGE: ${TOOLING_IMAGE}
  mssql:
    image: sitecore-dcrm-xp1-mssql:${SITECORE_VERSION}
    build:
      context: ./module/mssql
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-mssql:${SITECORE_VERSION}
        DEF_IMAGE: ${DEF_IMAGE}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
  xdbsearchworker:
    image: sitecore-dcrm-xp1-xdbsearchworker:${SITECORE_VERSION}
    build:
      context: ./module/xdbsearchworker
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-xdbsearchworker:$
{SITECORE_VERSION}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
        TOOLING_IMAGE: ${TOOLING_IMAGE}
```

If you are deploying XP0, add instructions for `xconnect`:

```
  xconnect:
  image: sitecore-dcrm-xp0-xconnect:${SITECORE_VERSION}
  build:
    context: ./module/xconnect/
    args:
      BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp0-xconnect:${SITECORE_VERSION}
      DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
      TOOLING_IMAGE: ${SITECORE_TOOLS_REGISTRY}sitecore-docker-tools-assets:$
{TOOLS_VERSION}
```

If you are deploying XP1, add instructions for `xdbsearch` and `xdbcollection`:

```
  xdbsearch:
    image: sitecore-dcrm-xp1-xdbsearch:${SITECORE_VERSION}
    build:
      context: ./module/xconnect
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-xdbsearch:${SITECORE_VERSION}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
        TOOLING_IMAGE: ${TOOLING_IMAGE}
  xdbcollection:
    image: sitecore-dcrm-xp1-xdbcollection:${SITECORE_VERSION}
    build:
      context: ./module/xconnect
      args:
```

```
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-xdbcollection:$
{SITECORE_VERSION}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
        TOOLING_IMAGE: ${TOOLING_IMAGE}
```

9. In the Windows console, go to the folder containing the `docker-compose.override.yml` file. Run the command `docker-compose build`.
10. After the build has completed, run the command `docker-compose up -d`.
11. When the Docker compose command has finished, rebuild your search indexes.

NOTE

Some modifications to Sitecore deployments, such as adding connection strings or changing the web configuration files, require you to use configuration transforms to change the configuration files. For information on how to apply configuration transforms, see the Sitecore container development documentation.

4. Add the DCRM connector module to Sitecore in Azure Kubernetes Service

To add the Sitecore Connect for Microsoft Dynamics 365 for Sales (DCRM) connector in Azure Kubernetes Service (AKS) you must do the following in this order:

- Build the DCRM images and push them to Azure.
- Prepare files and folders for deployment.
- Deploy the containers using *kubectl* commands.
- Update your Solr indexes.

4.1. Build images and push them to Azure

To build the images for DCRM and push them to Azure:

1. Build the images for DCRM as explained in Add the DCRM connector module to Sitecore in Docker.
2. Open the Windows console, and use the `docker tag` command to tag the images. For example:

```
docker tag sitecore-dcrm-xp1-cm:7.0.0-1497 $registry/experimental/sitecore-dcrm-xp1-cm:sc102
```

3. In the console, use the `docker push` command to push the images to your Azure registry. For example:

```
docker push $registry/sitecore-dcrm-xp1-cm:sc102
```

4.2. Prepare files and folders for deployment

To prepare files and folders in your installation for deployment:

1. Download the Sitecore DCRM container deployment package from the Sitecore download page and extract it to a folder on your local workstation.
2. Open the folder that you extracted the Sitecore DCRM container deployment package to.
3. Navigate to the `dcrm\k8s\<version>` folder, for example, `dcrm\k8s\ltsc2019`. Copy the `overrides` subfolder to the Sitecore Experience Platform (SXP) container deployment package folder `k8s\<version>` (on the same level as the *xp1* folder).

4. In the SXP container deployment package, in each of the `overrides\<topology>`, `overrides\<topology>\init`, and `overrides\<topology>\secrets` folders, locate the `kustomization.yaml` file. In each file, update the `bases` parameter with the appropriate folder names for your installation, for example, `../../xp1`.

   NOTE
   The `bases` parameter contains the placement of the original Sitecore container deployment files that the `kustomization.yaml` files override.

5. In each of the `kustomization.yaml` files, in the `images:` section, update the `newName` and `newTag` parameters with the values for the images you pushed to the Azure Registry, for example, `mssql-init`, `cm`, `xdbcollection`, `xdbsearch`, and `xdbsearchworker` .

   NOTE
   To find the values you need for, for example, the `mssql-init` image, go to the Azure Container Registry, search for your `sitecore-dcrm-xp1-mssql-init` image, and take the values from that image.

6. In the `overrides\xp1\secrets` folder, in the `sitecore-data-exchange-staging.txt` and `sitecore-dcrm.txt` files , update the connection string details.

   NOTE
   The files contain an example of how the connection string must look.

4.3. Deploy the containers

Prepare the AKS cluster configuration and deploy the ingress controller. For information on how to do this, see the *Installation Guide for Production Environment for Kubernetes* which is available on the Sitecore download page.

To deploy the containers and the necessary Kubernetes components:

1. Open the Windows console, and navigate to the folder containing the `xp1` and `overrides` folders.
2. Deploy the secrets. Use this command:

   ```
   kubectl apply -k ./overrides/<topology>/secrets/
   ```

3. Run the `external` folder. Use this command:

   ```
   kubectl apply -k ./<topology>/external/
   ```

4. Wait for all containers to have the status *Ok/Running*. You can check the status with this command:

```
kubectl get pods -o wide
```

5. Run the `init` folder. Use this command:

```
kubectl apply -k ./overrides/<topology>/init/
```

6. Wait for all containers to have the status *Completed*. You can check the status with this command:

```
kubectl get pods
```

7. To create persistent volumes, run this command:

```
kubectl apply -f ./xp1/volumes/azurefile
```

8. Run the Sitecore containers with the DCRM changes. Use this command:

```
kubectl apply -k ./overrides/<topology>/
```

9. Wait for all containers to have the status *Ok/Running*. You can check the status with the `kubectl get pods` command.
10. Update the local host file. For information on how to do this, see the *Installation Guide for Production Environment for Kubernetes*, which is available on the Sitecore download page.

When the containers have been deployed, rebuild your search indexes.

5. Upgrade from DCRM 6.0 to DCRM 7.0 on Docker

This section explains how to upgrade from Sitecore Connect for Microsoft Dynamics 365 for Sales (DCRM) 6.0 to DCRM 7.0 on a container platform.

5.1. Requirements

Before you perform the upgrade for DCRM container 6.0 to DCRM container 7.0, you must have the following:

- Sitecore Experience Platform (SXP) 10.1 deployed on Docker
- DCRM 6.0 deployed on Docker
- An up to date back up of the current DCRM 6.0 container files

5.2. Upgrade process

You must upgrade your SXP installation to 10.2 and your DCRM installation to 7.0 together.

To do so, you must:

1. Build new docker images for SXP 10.2 and DCRM 7.0.
2. Build an mssql-upgrade image
3. Perform the upgrade

5.2.1. Build new Docker images

To build the SXP 10.2 and DCRM 7.0 Docker images:

1. Download the SXP container deployment 10.2 package from the Sitecore download page. Extract it to a new folder on your local workstation with the folder structure intact. Name the new folder, for example, `DCRM7`.
2. Download the DCRM container deployment 7.0 package from the Sitecore download page. Extract it to your local workstation with the folder structure intact. Copy the files in the `dcrm\compose\<windows version>\<topology>` folder and paste them into the `\compose\<windows version>\<topology>` folder in the SXP 10.2 deployment structure.
3. Navigate to your Sitecore 10.1 container deployment folder. Copy the databases from the `mssql-data` folder and paste them in the Sitecore 10.2 container deployment `mssql-data` folder.
4. To deploy Sitecore 10.2, open a PowerShell window with administrator rights, navigate to the SC 10.2 Container deployment folder, and run the following commands:

```
docker-compose build
docker-compose up
```

5. Verify that Sitecore Container 10.2 is up and running successfully.

5.2.2. Build the mssql-upgrade image

You use a custom mssql-upgrade image to upgrade a Sitecore solution that has DCRM installed. To build a custom mssql-upgrade image, you must download the latest Sitecore XP mssql-upgrade image from the container registry and create a custom Dockerfile on top of it.

1. From the **Resource files for Modules 1.0.0** section on the Sitecore download page, download the Dynamics CRM Connect Upgrade resources 1.0.0 package. Extract it to a folder on your local machine.
2. From the upgrade resources, copy the `Dynamics CRM Connect Upgrade resources 1.0.0\6.0.0\Data` folder and paste it into the upgrade folder for the Windows version and topology you are using, for example, `ltsc2019\upgrade\xp1`.
3. In the `DCRM Connect Upgrade resources 1.0.0\6.0.0` folder, create a docker file and name it `Dockerfile`. In the file, add instructions to point its base image to the 10.2 mssql-upgrade image. The file will look, for example, like this:

```
ARG BASE_IMAGE= ${SITECORE_DOCKER_REGISTRY} /sxp/sitecore-xp1-mssql-upgrade : tag
FROM ${BASE_IMAGE}
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
# Add ? DCRM?  module
COPY <Data folder local path> "C:\data\ResourceItems\10.1\modules"
```

> **NOTE**
>
> In the Dockerfile, ensure that ARG BASE_IMAGE parameter points to the mssql-upgrade image, and that the Data folder local path is set to the 6.0.0\Data folder in the folder you created in Step 1.

4. To build the mssql-upgrade image, open a PowerShell window, navigate to the folder where you placed the Dockerfile, and run the following command:

```
docker build . -t "<imageName>:<available port number>”
```

5. Verify that Docker has created an image with the name you specified.

5.2.3. Perform upgrade process

To perform the upgrade:

1. On your local machine, in a PowerShell window, navigate to the Sitecore Container Deployment 10.2 folder. Navigate to the upgrade folder for the Windows version and topology you are using, for example, `ltsc2019\upgrade\xp1`.
2. In the topology folder, run the `compose-init.ps1 script`. This script updates the environment configuration file with the appropriate values for all the environment variables including the SQL username, SQL password, SQL Server address, and the Sitecore license file.

> **NOTE**
> For more information about running the script to prepare for the deployment, see the Installation Guide for Developer Workstation with Containers on the Sitecore download page.

3. In the upgrade folder, verify that the `upgrade.env` file has the correct details. For more details, see the Upgrade Container Deployment Guide for SXP 10.2.0.
4. In the `docker-compose.upgrade.yml` file, update the `image` setting with the mssql-upgrade image you created previously.
5. Open a new PowerShell window with administrator rights. Navigate to the upgrade directory.
6. Verify that the mssql container is up and running.
7. To perform the upgrade, run this command:

```
docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env up
```

8. To check the status of the upgrade, run this command:

```
docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env ps
```

9. When the upgrade process is completed, you can clean up your environment.
   If you ran the upgrade container in Docker Compose, from the Docker Compose folder for the topology that you upgraded, for example, `sitecore-xp1`, run the following PowerShell cmdlet:

```
docker-compose.exe -f .\docker-compose.upgrade.yml --env-file .\upgrade.env down
```

6. Upgrade from DCRM 6.0 to DCRM 7.0 on Kubernetes

This section explains how you upgrade DCRM Container in AKS from version 6.0 to 7.0.

6.1. Requirements

Before you perform the upgrade for Container DCRM Container 6.0 to DCRM 7.0, you must:

- Have a Sitecore AKS 10.1 deployment with DCRM 6.0
- Back up the 10.1 database.

6.2. Build and push the mssql-upgrade image

To upgrade DCRM you must build an mssql-upgrade image. To do so:

1. Build the images for DCRM as explained in the *Build the mssql-upgrade image* section in Upgrade from DCRM 6.0 to DCRM 7.0 on Docker.
2. In a PowerShell window, navigate to the folder containing the Dockerfile and build and tag the image. For example:

```
docker build . -t sitecore-dcrm-xp1-cm:<imageVersionTag> $registry/sitecore-dcrm-xp1-cm:<newTag>
```

3. Check that Docker has created the image.
4. Push the created image to your Azure registry. For example:

```
docker push $registry/sitecore-dcrm-xp1-cm:<newTag>
```

6.3. Perform upgrade process

To upgrade to DCRM 7.0 on Sitecore 10.2:

1. Download and extract Sitecore 10.2 Sitecore Container Deployment Package from the Sitecore download page. Extract it to your local workstation with the folder structure intact.
2. Navigate to the upgrade folder for the Windows version and topology you are using, for example, `k8s\ltsc2019\upgrade\xp1`. In the `kustomization.yaml` file, update the

images section with `newName` and `newTag` of the custom mssql-upgrade image you created and pushed previously.

3. In the `configuration` folder, update the secrets files. For more information on the secrets files, please refer to the *Installation Guide for Production Environment with Kubernetes* guide available on the Sitecore download page.
4. Download the DCRM container deployment 7.0 package from the Sitecore download page. Extract it to your local workstation with the folder structure intact. Copy the `dcrm\k8s\<windows version>\overrides` folder and paste it into the `\k8s\<windows version>\` folder in the SXP 10.2 deployment structure.
5. In the DCRM `secrets` folder in the SXP 10.2 structure, update these secrets files:
   - `sitecore-data-exchange-staging.txt`
   - `sitecore-dcrm.txt`

   NOTE
   The DCRM secrets folder in the SXP 10.2 structure will, for example, be `k8s\ltsc2019\overrides\xp1\secrets`.
6. Log in to the Azure CLI and set a subscription.

```
az login
az account set --subscription "Your Subscription"
```

7. Get the credentials for the Kubernetes cluster that was created with the AKS cluster.

```
zaks get-credentials --resource-group <10.1 resource group>--name <10.1 cluster>
```

8. To deploy the Sitecore upgrade job, go to the folder where the updated files are, for example `k8s\ltsc2019\upgrade\xp1`, and run this command:

```
kubectl apply -k .\
```

9. To check if the job has completed, run this command:

```
kubectl get pod
```

10. When the upgrade process is completed, you can delete the Kubernetes upgrade job and upgrade secrets. In the console, go to the folder you used in step 2. and run these commands:

```
kubectl delete -f .\
kubectl delete -k .\
```

NOTE

Upgrading with a custom mssql-image includes cleaning up DCRM items from the database since DCRM 7.0 has moved to using resources files. This requires rebuilding and upgrading the Sitecore role instances cm, xconnect and xdbsearchworker.

For detailed instructions on how to build and deploy the Sitecore role instances please refer to the Sitecore *Upgrade Container Deployment Guide* on the Sitecore download page.

7. Rebuild the search indexes

When you have deployed the containers, you must rebuild your search indexes.

To rebuild the indexes:

1. Browse to your Sitecore URL, for example, `https://xp1cm.localhost/`. Open the control panel.
2. In the **Indexing** section, click **Populate Solr Managed Schema**.
3. In the **Schema Populate** dialog box, click **Select All**, then click **Populate**. Wait for the process to finish.
4. On the Control Panel, in the **Indexing** section, click **Indexing Manager**. In the **Indexing Manager** dialog, select the following indexes:
    - sitecore_master_index
    - sitecore_core_index
    - sitecore_web_index
    - sitecore_marketingdefinitions_master
    - sitecore_marketingdefinitions_web
5. Click **Rebuild**. When the indexes have been rebuilt, click **Close**.
6. Open the Content Editor with Master as the content database.
7. Navigate to */sitecore/system/Data Exchange*. On the **Folder** tab, verify that the **Empty Data Exchange Tenant** and **DCRM Tenant** buttons are available.