Sitecore Connect for Microsoft Dynamics 365 for Sales 6.0 Container Deployment Guide

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

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 site.
- 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 site.

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 Developer Portal. 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. Add the following variables to the `.env` file:

```
DEF_IMAGE=scr.sitecore.com/sxp/modules/sitecore-def-xp1-assets:6.0.0.0-1809
DCRMCNN_IMAGE=scr.sitecore.com/sxp/modules/sitecore-dcrm-xp1-assets:6.0.0.0-1809
TOOLING_IMAGE=scr.sitecore.com/tools/sitecore-docker-tools-assets:10.1.0-1809
```

7. Save the `.env` file.
8. 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:
   - `mssql`
   - `cm`
   - `xconnect`
   - `xdbsearchworker`
   - `mssql-init`

   > **NOTE**
   > The `mssql-init` image is only necessary if you are deploying to Azure Kubernetes Services (AKS).

3. In each subfolder, create a new file and name it `Dockerfile`.
4. In the `mssql` folder, in the `Dockerfile` file, enter the following instructions:

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

FROM ${DEF_IMAGE} as def
FROM ${DCRMCNN_IMAGE} as dcrm
FROM ${BASE_IMAGE}

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

# Deploy DEF dacpac file
COPY --from=def C:\module\db C:\def_data
RUN C:\DeployDatabases.ps1 -ResourcesDirectory C:\def_data; `
    Remove-Item -Path C:\def_data -Recurse -Force;

# Deploy DCRM dacpac file
COPY --from=dcrm  C:\module\db C:\dcrm_data
RUN C:\DeployDatabases.ps1 -ResourcesDirectory C:\dcrm_data; `
    Remove-Item -Path C:\dcrm_data -Recurse -Force;
```

5. 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
```

6. 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
```

7. 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
```

8. If you are deploying to AKS, in the `mssql-init` folder, in the `Dockerfile` file, enter the following instructions:

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

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

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

# Deploy DEF dacpac file
COPY --from=def C:\module\db C:\resources\def_data

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

9. 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}
```

If you are deploying XP1 to AKS, add instructions for `mssql-init`:

```
  mssql-init:
    image: sitecore-dcrm-xp1-mssql-init:${SITECORE_VERSION}
    build:
      context: ./module/mssql-init
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp1-mssql-init:${SITECORE_VERSION}
        DEF_IMAGE: ${DEF_IMAGE}
        DCRMCNN_IMAGE: ${DCRMCNN_IMAGE}
```

10. In the Windows console, go to the folder containing the `docker-compose.override.yml` file. Run the command `docker-compose build`.
11. After the build has completed, run the command `docker-compose up -d`.

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.

3.3. Update the Solr indexes

When the Docker compose command has finished, you must update your Solr indexes.

To update the indexes:

1. When the Docker compose command finishes, browse to your Sitecore URL, for example, `https://xp0cm.localhost/`. Open the control panel and click **Populate Solr Managed Schema**.
2. After Sitecore has populated the Solr Schema, click **Indexing Manager**.
3. In the **Indexing Manager** dialog, select the indexes you want to update, and click **Rebuild**. When the indexes have been rebuilt, click **Close**.

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.

   > **NOTE**
   > The Kubernetes deployment requires an `mssql-init` image. You must ensure that you include `mssql-init` in your `docker-compose-override.yml` file.

2. Open the Windows console, and use the `docker tag` command to tag the images. For example:

   ```
   docker tag sitecore-dcrm-xp1-cm:6.0.0.0-1809 $registry/experimental/sitecore-dcrm-xp1-cm:sc101
   ```

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:sc101
   ```

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 Developer Portal 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 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.

4.4. Update Solr indexes

To update your Solr indexes:

1. Browse to your Sitecore URL, for example, `https://cm.globalhost/`. Open the control panel and click **Populate Solr Managed Schema**.
2. After Sitecore has populated the Solr Schema, click **Indexing Manager**.
3. In the **Indexing Manager** dialog, select the indexes you want to update, and click **Rebuild**. When the indexes have been rebuilt, click **Close**.