Data Exchange Framework 7.0 container deployment guide

A guide to deploying Sitecore Data Exchange Framework to Docker or Azure Kubernetes Services

November 29, 2021

Table of Contents

1. Introduction to Data Exchange Framework for containers

Sitecore Data Exchange Framework enables you to synchronize data between Salesforce and third-party systems.

This guide shows you how to add the Data Exchange Framework to Sitecore container installations for Docker and Azure Kubernetes Service.

2. Prepare to deploy DEF to Sitecore containers

This section explains what you need to prepare for deploying the Sitecore Data Exchange Framework (DEF) to Sitecore containers for Docker and Azure Kubernetes Service (AKS).

2.1. Requirements

Before you deploy DEF to Docker or AKS, the following requirements must be met:

- Docker Desktop must be 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 Downloads site.
- If the installation is done on Kubernetes, you must 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 Downloads site.

2.1.1. Sitecore topologies, containerized roles and the DEF module

For each Sitecore deployment topology, the following table lists the containerized Sitecore roles where you deploy the DEF module. The information applies to both Docker and Kubernetes deployments.

Containerized role	**Sitecore topology and the DEF module**		
	XP1	**XP0 (Docker only)**	**XM1**
CD (content delivery)	Yes	not applicable	not applicable
CM (content management)	Yes	Yes	Yes
ID (Identity service)	Yes	Yes	not applicable
xdbautomationworker	Yes	Yes	not applicable

3. Add Data Exchange Framework module to Sitecore in Docker

To add the Sitecore Data Exchange Framework (DEF) module 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 DEF container deployment package from the Sitecore Downloads page. Extract it to your local workstation with the folder structure intact.
2. In the folder where you extracted the DEF container deployment package, open the folder for the Windows version and topology you are using, for example, `def\compose\ltsc2019\xp1`.
3. Open the `.env-example` file in an editor. The following shows an example of what the contents of the file looks like:

   ```
   TOPOLOGY=xp1
   #Note: Copy this below if you are using the tenant service in DEF
   TENANT_SERVICE_HOST=xp1ts.localhost
   ```

4. Copy all the required variables in the file to the clipboard.
5. Go to the Sitecore 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`.
6. Open the `.env` file in an editor, and paste in the variables from the DEF `.env-example` file.
7. Add the following variables to the `.env` file:

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

8. Save the `.env` file.
9. From the DEF `compose\<version>\<topology>` folder, copy the `docker-compose.override.yml` file and paste it to the Sitecore container deployment `compose\<version>\<topology>` folder (where the `docker.compose.yml` file is).
10. If you are not using the tenant service in DEF, open the `docker-compose.override.yml` file, and remove the sections for the `xdbautomationworker` and `id` images.

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 Sitecore 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`
   - `id`
   - `xdbautomationworker`

   > **NOTE**
   > You only need the `xdbautomationworker` and `id` modules if you are using the tenant service. If you are not using the tenant service, you do not have to create the folders and files for these two modules.
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 DEF_IMAGE
ARG TOOLING_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE}

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

WORKDIR C:\inetpub\wwwroot

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

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

5. If you are using the tenant service, in the `id` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`

ARG BASE_IMAGE
ARG DEF_IMAGE
ARG TOOLING_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE}

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

WORKDIR C:\Identity

# Add DEF module
COPY --from=def \module\transforms\ C:\transforms\

COPY --from=tooling \tools\ \tools\
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\Identity -XdtPath c:\transforms\id
```

6. If you are using the tenant service, in the `xdbautomationworker` folder, in the `Dockerfile` file, enter the following instructions:

```
# escape=`

ARG BASE_IMAGE
ARG DEF_IMAGE
ARG TOOLING_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${TOOLING_IMAGE} as tooling
FROM ${BASE_IMAGE}

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

# Add DEF MA module
COPY --from=def \module\xdbautomationworker\content C:\service\
COPY --from=def \module\transforms\C:\transforms\
COPY --from=tooling \tools\\tools\
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\service -XdtPath \transforms
\xdbautomationworker
```

7. In the `compose\<version>\<topology>\docker-compose.override.yml` file, add build instructions for each role.
   For all topologies, you must add instructions for the `cm` role, for example:

```
services:
   cm:
     image: sitecore-def-${TOPOLOGY}-cm:${SITECORE_VERSION}
     build:
       context: ./module/cm
       args:
         BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-${TOPOLOGY}-cm:${SITECORE_VERSION}
         DEF_IMAGE: ${DEF_IMAGE}
         TOOLING_IMAGE: ${TOOLING_IMAGE}
```

   If you are deploying the XP1 topology, add instructions for the `cd` role as follows:

```
cd:
  image: sitecore-def-${TOPOLOGY}-cd:${SITECORE_VERSION}
```

```
  build:
     context: ./module/cm
     args:
       BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-${TOPOLOGY}-cd:${SITECORE_VERSION}
       DEF_IMAGE: ${DEF_IMAGE}
       TOOLING_IMAGE: ${TOOLING_IMAGE}
```

If you are deploying XP0 or the XP1 topology, add instructions for the `id` and the `xdbautomationworker` roles, for example:

```
id:
  image: sitecore-def-${TOPOLOGY}-id:${SITECORE_VERSION}
  build:
     context: ./module/id
     args:
       BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-id:${SITECORE_VERSION}
       DEF_IMAGE: ${DEF_IMAGE}
       TOOLING_IMAGE: ${TOOLING_IMAGE}
xdbautomationworker:
  image: sitecore-def-${TOPOLOGY}-xdbautomationworker:${SITECORE_VERSION}
  build:
      context: ./module/xdbautomationworker
     args:
       BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-${TOPOLOGY}-xdbautomationworker:$
{SITECORE_VERSION}
       DEF_IMAGE: ${DEF_IMAGE}
       TOOLING_IMAGE: ${TOOLING_IMAGE}
```

8. In the Windows console, go to the folder containing the `docker-compose.override.yml` file. Run the following command:

```
docker-compose build
```

9. When the build completes, run the following command:

```
docker-compose up -d
```

10. 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 Data Exchange Framework module to Sitecore in Azure Kubernetes Service

To add Data Exchange Framework (DEF) in Azure Kubernetes Service (AKS) you must do the following in this order:

- Build images and push them to Azure
- Prepare configuration files for deployment
- Deploy the containers
- Update Solr indexes

4.1. Build images and push them to Azure

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

1. Prepare the installation files as explained in Add Data Exchange Framework module to Sitecore in Docker.
2. Build the images for DEF as explained section *Build the Docker images*, in Add Data Exchange Framework module to Sitecore in Docker.
3. Tag the images with the `docker tag` command. For example:

```
docker tag sitecore-def-xp0-assets:6.0.0.01525.109-10.0.19042.804-2009 $registry/
experimental/def/sitecore-xp1-cm:sc101def
```

4. Push the images to your Azure registry with the `docker push` command. For example

```
docker push $registry/experimental/def/sitecore-xp1-cm:sc101def
```

4.2. Prepare configuration files for deployment

To prepare configuration files in your installation for deployment:

1. Open the folder where you extracted the Data Exchange Framework container deployment package.
2. Navigate to the `DEF.Asset\k8s\<version>` folder, for example, `DEF.Asset\k8s\ltsc2019`. Copy the `overrides` subfolder to the Sitecore Experience Platform (SXP) container deployment package, in the `k8s\<version>` folder (on the same level as the `xp1` folder).

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

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

4. In each `kustomization.yaml` file, in the `images:` section, update the `newName` and `newTag` parameters with the values for the images you pushed to the Azure Registry, for example `cm`, `cd`, `xdbautomationworker`, and `id`.

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

5. If you are deploying the XP1 topology, in the `overrides\<topology>\secrets` folder, in the `sitecore-tenant-service-connection-string.txt` file, update the connection string details. The file contains an example of how the connection string should look.

4.3. Deploy the containers

Before you deploy the containers to Kubernetes, you must 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 `k8s\<version>` folder.
2. Deploy the secrets:
   - If you are using the XP1 topology, run the following command:

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

   - If you are using the XM1 topology, run the following command:

     ```
     kubectl apply -k ./<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 ./<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 ./<topology>/volumes/azurefile
```

8. Run the Sitecore containers with the SFCRM 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. Obtain the external IP address. Use this command:

```
kubectl get service -l app=nginx-ingress
```

11. 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. 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 (if you are using the Marketing Automation plugin)
    - sitecore_marketingdefinitions_web (if you are using the Marketing Automation plugin)
5. Click **Rebuild**. When the indexes have been rebuilt, click **Close**.

6. Upgrade the Data Exchange Framework connector to version 7.0 in Docker

This section explains how you upgrade Data Exchange Framework (DEF) Container 6.0 to DEF 7.0. DEF is compatible with Sitecore 10.2.

6.1. Requirements

- Back up the current DEF 6.0 container files.

6.2. Build the Docker images for Sitecore 10.2 Container with DEF 7.0.

To upgrade DEF, you must build new Docker images. To do so:

1. Download the Sitecore container deployment package for 10.2 from the Sitecore download page. Extract it to your local workstation with the folder structure intact. Name the folder, for example *DEF 102*.
2. Download the DEF container deployment package for 7.0 package from the Sitecore download page. Extract it to your local workstation, in the folder where you extracted the Sitecore 10.2 container package, with the folder structure intact. For example, copy it to the *DEF 102* folder.
3. Go to the current Sitecore 10.1 Container deployment folder, copy the databases from `mssql-data` folder, and paste it in Sitecore 10.2 `mssql-data` folder. For example, copy it to `c:/DEF102/mssql-data`.
4. Open PowerShell window with the Run as Administrator option, go to the Sitecore 10.2 Container directory, and to deploy Sitecore 10.2 container, run the following commands:
    - Run the command `docker-compose build`.
    - Run the command `docker-compose up`
5. Make sure that Sitecore Container 10.2 is up and and running successfully.

6.3. Build the mssql-upgrade image

To upgrade a Sitecore solution that has DEF installed, you must build a custom `mssql-upgrade` image. To do so:

1. On the download page for Resource files for Modules, download the DEF Module Upgrade Resource file, and extract it to your local workstation with the structure intact.
2. 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, for example, will look like this:

```
ARG BASE_IMAGE=ideftdevacr.azurecr.io/sxp/sitecore-xp1-mssql-
upgrade:10.2.0.006572.994-10.0.17763.2183-ltsc2019-unstable
FROM ${BASE_IMAGE}
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference =
'SilentlyContinue';"]
# Add DEF module
COPY <DEF Module Upgrade Resource file path> "C:\data\ResourceItems\10.2.0\modules\"
```

> **NOTE**
> In the Dockerfile, ensure that the `ARG BASE_IMAGE` value is pointing to Sitecore 10.2 mssql-upgrade image.

3. In a PowerShell window, to the same path as the Docker file, and run the following command:

```
docker build . -t "<imageName>:<AnyAvailablePortNo>" command
```

   For example: `docker build . -t "upgradedefimage:3636`.

4. Ensure that the image has been created.

6.3.1. Perform the upgrade

To perform the upgrade:

1. In your local machine, in a PowerShell window, navigate to the DEF container deployment folder. Navigate to the upgrade folder for the Windows version and topology you are using, for example `Compose/<version>/upgrade/<topology>`.
2. Navigate to the `Compose/<version>/upgrade/<topology>` folder, and 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 Sitecore Installation Guide for Developer Workstation with Containers on the Sitecore download page.

3. In the same directory as the `compose-init.ps1` script, open the `compose.upgrade.yml` file, and the new image value (for example, `image:upgradedefimage:3636`)
4. Ensure that the mssql container is up and running.

5. To perform the upgrade, run the following command:

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

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

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

7. 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, run the following PowerShell cmdlet:

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

7. Upgrade the DEF connector to version 7.0 in Kubernetes

This section explains how you can upgrade Data Exchange Frame Container (DEF) 6.0 to DEF 7.0. DEF 7.0 is compatible with Sitecore 10.2.

7.1. Requirements

Before you upgrade DEF for container 6.0 to DEF for container 7.0 you must:

- Have an existing Sitecore 10.1 Kubernetes deployment with DEF connector.
- Back up your current mssql databases.

7.2. Build and push the mssql-upgrade image

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

1. Build the images for DEF as explained in the *Build the mssql-upgrade image* section in Upgrade the Data Exchange Framework connector to version 7.0 in Docker.
2. Open a PowerShell window, go to the same path as the Dockerfile, and run the following command:

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

3. Check that the image has been created:
4. Run the following command to push the image you created to your Azure registry. For example:

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

7.3. Perform upgrade process

To upgrade to DEF 7.0 on Sitecore 10.2:

1. Download the Sitecore container deployment package for 10.2, and 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 DEF container deployment package for 7.0 package from the Sitecore download page. Extract it to your local workstation with the folder structure intact. Copy the `DEF.Asset\k8s\<windows version>\overrides` folder and paste it into the `\k8s\<windows version>\` folder in the SXP 10.2 deployment structure.
5. In the `secrets` folder in the SXP 10.2 structure, for example, `k8s\ltsc2019\overrides\xp1\secrets`, update the `sitecore-tenant-service-connection-string.txt` secret file.
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 by running this command:

```
az aks 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 DEF items from the database since DEF 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.