Sitecore Connect for Salesforce CRM 6.0 Container Deployment Guide

Table of Contents

1. Introduction

Sitecore Connect for Salesforce CRM (SFCRM) enables you to synchronize data between Salesforce CRM and the Sitecore Experience Platform.

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

2. Prepare the installation

This section explains what you need to install the Sitecore Connect for Salesforce CRM (SFCRM) connector in Sitecore containers for Docker and Azure Kubernetes Service.

2.1. Requirements

Before you add the SFCRM 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.
- Access to a Salesforce CRM instance and a user account on Salesforce. This account must at the least have access to read data from Salesforce. If you are moving data to from Sitecore to Salesforce, the account must also have rights to write data to Salesforce.

To prepare for the installation, you must:

- prepare a Salesforce connection string to Sitecore.

2.2. Prepare a Salesforce connection string to Sitecore

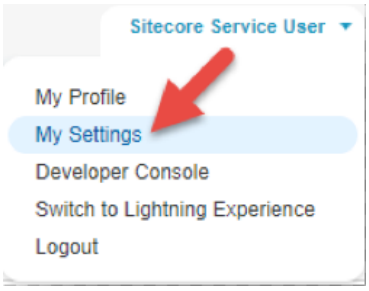
To construct a Salesforce connection string to your Sitecore installation you need to obtain some values from Salesforce. To obtain these values:

1. In Salesforce, make a note of the following values:
    - *User ID* - the ID that Sitecore uses to call the Salesforce API. This user ID does not need administrator rights, but it must have sufficient rights to perform the activities you want to perform from Sitecore. For example:
        - To have Salesforce contacts created in Sitecore, the user must have read-access on contacts and campaigns.
        - To push contact data from Sitecore into Salesforce, the user must have write-access on contacts.

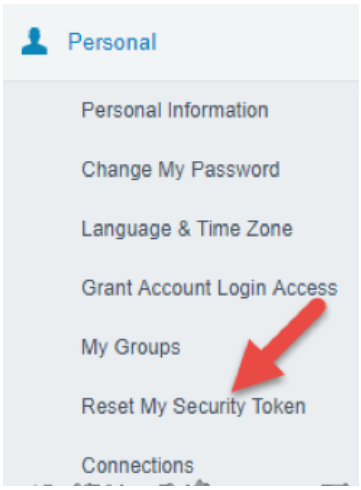
- *Password* - the password for the Salesforce user that Sitecore uses to call the Salesforce API.

> **NOTE**
> The password must not contain the semicolon (;) character.

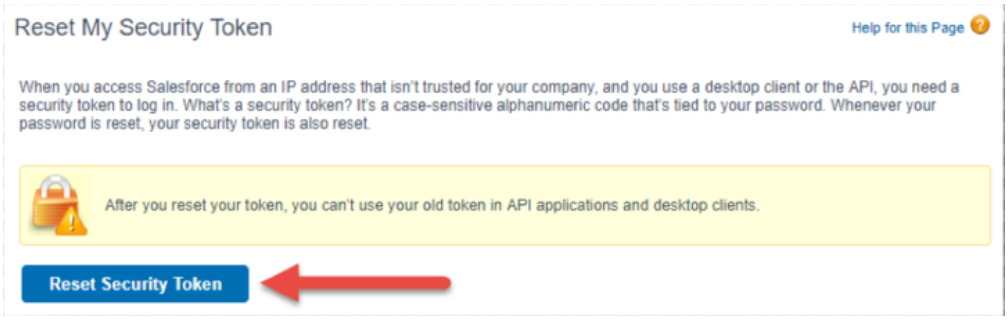
2. To obtain the *security token value*, log in to Salesforce with the user ID.
3. In the top menu, click the user name, then click **My Settings**.



4. In the left menu, click **Personal**, then click **Reset My Security Token**.

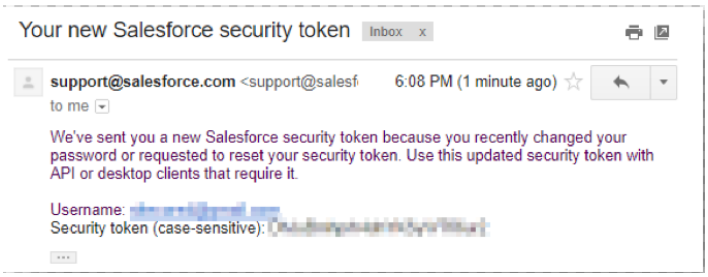


5. Read the warning about resetting security tokens. If you are ready to proceed, click **Reset Security Token**.

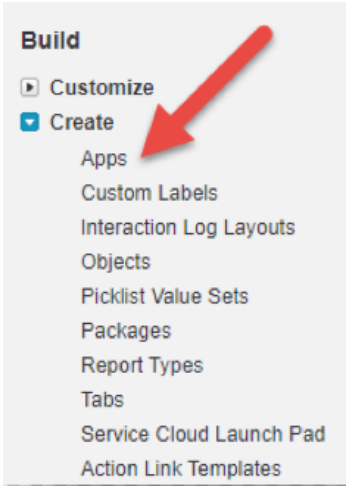


Salesforce informs you that the new security token will be emailed to you.

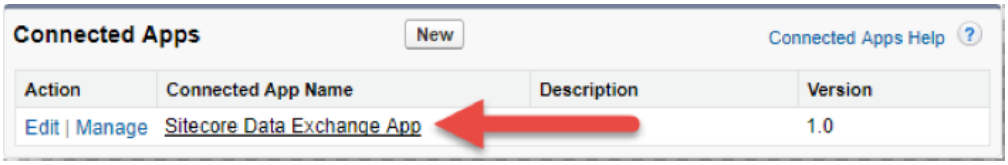
6. In your inbox, find the email message from Salesforce. Locate the *Security token (case-sensitive)* line, and make a note of the new security token value.



7. To obtain the *Consumer Key* and *Consumer Secret* values, in Salesforce, in the left menu, in the **Build** section, expand **Create** and then click **Apps**.



8. In the **Connected Apps** section, click the name of the connected app you created. For information on how to create a connected app, see the Sitecore Connect for Salesforce CRM 6.0 Installation Guide on the Sitecore download page.



9. In the **API (Enable OAuth Settings)** section, copy the values for the *Consumer Key* and *Consumer Secret*.

10. Use the values to construct a connection string with this format:

```
MYSF_CONNECTIONSTRING=user id=<user id>;
                      password=<password>;
                      client id=<Consumer Key>;
                      secret key=<Consumer Secret>;
                      security token=<security token value>;
```

You will need this connection string when you add the SFCRM module to Docker or Kubernetes.

3. Add the SFCRM connector module to Sitecore in Docker

To add Sitecore Connect for Salesforce CRM (SFCRM) 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 SFCRM 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 SFCRM container deployment package to. Go to the folder for the Windows version and topology you are using, for example, `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 SFCRM `.env-example` file. Replace the default value for `MYSF_CONNECTIONSTRING` with the connection string you prepared in Prepare the installation.
6. From the SFCRM `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 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 folder and name it `module`.
2. In the `module` folder, create these subfolders:
   - `mssql`
   - `mssql-init`
   - `cm`
   - `xconnect`
   - `xdbsearchworker`
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 SFCRM_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${SFCRM_IMAGE} as sfcrm
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 SFCRM dacpac file
COPY --from=sfcrm C:\module\db C:\sfcrm_data
RUN C:\DeployDatabases.ps1 -ResourcesDirectory C:\sfcrm_data; `
    Remove-Item -Path C:\sfcrm_data -Recurse -Force;
```

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

```
# escape=`
ARG BASE_IMAGE
ARG DEF_IMAGE
ARG SFCRM_IMAGE

FROM ${DEF_IMAGE} as def
FROM ${SFCRM _IMAGE} as sfcrm
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 SFCRM dacpac file
COPY --from=sfcrm C:\module\db C:\resources\sfcrm_data
```

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

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

FROM ${DEF_IMAGE} as def
FROM ${SFCRM_IMAGE} as sfcrm
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\
FROM baseImage as defsfcrm

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

# Add SFCRM module
COPY --from=sfcrm \module\cm\content C:\inetpub\wwwroot

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

# Add SFCRM connection strings in Sitecore config file
RUN C:\tools\scripts\Invoke-XdtTransform.ps1 -Path C:\inetpub\wwwroot -XdtPath
C:\transforms\role

#Enable loadUserProfile to allow CMS to connect to salesforce
RUN C:\windows\System32\inetsrv\appcmd.exe set config -section:system.applicationHost/
applicationPools /applicationPoolDefaults.processModel.loadUserProfile:"true" /
commit:apphost;
```

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

```
# escape=`
ARG BASE_IMAGE
ARG SFCRM_IMAGE
ARG TOOLING_IMAGE

FROM ${SFCRM_IMAGE} as sfcrm
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=sfcrm \module\xconnect\content C:\inetpub\wwwroot

#Copy transformation files
COPY --from=sfcrm \module\xdttransform\xconnect\transforms\ C:\transforms\role

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

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

```
# escape=`
ARG BASE_IMAGE
ARG SFCRM_IMAGE
ARG TOOLING_IMAGE

FROM ${SFCRM_IMAGE} as sfcrm
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=sfcrm \module\xdbsearchworker\content C:\service\

#Copy transformation files
COPY --from=sfcrm \module\xdttransform\xdbsearchworker\transforms\ C:\transforms\role

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

9. In the `compose\<version>\<topology>\docker-compose.override.yml` file, add build instructions for each role. If you are using, for example, the XP0 topology, the file will look like this:

```
services:
  cm:
    image: sitecore-sfcrm-xp0-cm:${SITECORE_VERSION}
    build:
      context: ./module
      dockerfile: ./cm/Dockerfile
      args:
        BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp0-cm:${SITECORE_VERSION}
        DEF_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-def-xp0-assets:$
{DEF_VERSION}
        SFCRM_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-sfcrm-xp0-assets:$
{SFCRM_VERSION}
        TOOLING_IMAGE: ${SITECORE_TOOLS_REGISTRY}sitecore-docker-tools-assets:$
{TOOLS_VERSION}
        environment:
          Sitecore_ConnectionStrings_mysf: ${MYSF_CONNECTIONSTRING}
mssql:
  image: sitecore-sfcrm-xp0-mssql:${SITECORE_VERSION}
  build:
    context: ./module
    dockerfile: ./mssql/Dockerfile
    args:
      BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp0-mssql:${SITECORE_VERSION}
      DEF_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-def-xp0-assets:${DEF_VERSION}
      SFCRM_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-sfcrm-xp0-assets:$
{SFCRM_VERSION}
xdbsearchworker:
  image: sitecore-sfcrm-xp0-xdbsearchworker:${SITECORE_VERSION}
  build:
    context: ./module
    dockerfile: ./xdbsearchworker/Dockerfile
    args:
      BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp0-xdbsearchworker:$
```

```
{SITECORE_VERSION}
      SFCRM_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-sfcrm-xp0-assets:$
{SFCRM_VERSION}
      TOOLING_IMAGE: ${SITECORE_TOOLS_REGISTRY}sitecore-docker-tools-assets:$
{TOOLS_VERSION}
xconnect:
  image: sitecore-sfcrm-xp0-xconnect:${SITECORE_VERSION}
  build:
    context: ./module
    dockerfile: ./xconnect/Dockerfile
    args:
      BASE_IMAGE: ${SITECORE_DOCKER_REGISTRY}sitecore-xp0-xconnect:${SITECORE_VERSION}
      SFCRM_IMAGE: ${SITECORE_DOCKER_REGISTRY}modules/sitecore-sfcrm-xp0-assets:$
{SFCRM_VERSION}
      TOOLING_IMAGE: ${SITECORE_TOOLS_REGISTRY}sitecore-docker-tools-assets:$
{TOOLS_VERSION}
```

10. In the `compose\<version>\<topology>\.env` file, add the asset image version. For example:

```
DEF_VERSION=<image version for your topology>
SFCRM_VERSION=<image version for your topology>
SITECORE_TOOLS_REGISTRY=scr.sitecore.com/tools/
TOOLS_VERSION=<image version for your topology>
```

> **NOTE**
> You can find the image version in the Sitecore Docker Images repository.

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

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. Open the Content Editor with *Master* as the content database.
4. In the content tree, navigate to `/sitecore/system/DataExchange`. On the **Folder** tab, verify that the **Empty Data Exchange Tenant** and **Connect for Salesforce Tenant** buttons are available.

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

To add the Sitecore Connect for Salesforce CRM (SFCRM) connector in Azure Kubernetes Service (AKS) you must do the following in this order:

- Build the SFCRM 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 SFCRM and push them to Azure:

1. Build the images for SFCRM as explained in Add the SFCRM 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-sfcrm-xp1-cm:10.1.0.005207.643-10.0.17763.1757-ltsc2019 $registry/
   sitecore-sfcrm-xp1-cm:<tag version>
   ```

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

   ```
   docker push $registry/sitecore-sfcrm-xp1-cm:<tag version>
   ```

4.2. Prepare files and folders for deployment

To prepare files and folders in your installation for deployment:

1. Download the Sitecore SFCRM 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 SFCRM container deployment package to.
3. Navigate to the `k8s\<version>\<topology>` folder, for example, `k8s\ltsc2019\xp1`. 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`, `overrides\xp1\init`, and `overrides\xp1\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 `mssql-init`, `cm`, `xdbcollection`, `xdbsearch` and `xdbsearchworker` images you pushed to the Azure Registry.
6. In the `overrides\xp1\secrets` folder, in the `sitecore-salesforce-crm-connection-string.txt` file , replace the content with the connection string you prepared in Prepare the installation.

4.3. Deploy the containers

To deploy the containers and the necessary Kubernetes components:

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

   ```
   kubectl apply -k ./overrides/xp1/secrets/
   ```

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

   ```
   kubectl apply -k ./xp1/external/
   ```

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

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

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

   ```
   kubectl apply -k ./overrides/xp1/init/
   ```

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

```
kubectl get pods
```

8. To create persistent volumes, run this command:

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

9. Run the Sitecore containers with the SFCRM changes. Use this command:

```
kubectl apply -k ./overrides/xp1/
```

10. Wait for all containers to have the status *Ok/Running*. You can check the status with the `kubectl get pods` command.
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.

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. Open the Content Editor with *Master* as the content database.
4. In the content tree, navigate to `/sitecore/system/DataExchange`. On the **Folder** tab, verify that the **Empty Data Exchange Tenant** and **Connect for Salesforce Tenant** buttons are available.