Sitecore Experience Commerce 10.2 Container Upgrade Guide

Instructions for upgrading a containerized environment from SXC 10.1 to SXC 10.2

November 4, 2021

Table of Contents

1. Introduction

This guide describes how to upgrade a Sitecore Experience Commerce™ (XC) 10.1 containerized solution to Sitecore XC 10.2.

Sitecore Experience Commerce™ (XC) 10.2 introduces a new Sitecore XC upgrade container that facilitates the XC databases upgrade process. Although the primary purpose of the Sitecore XC upgrade container is to facilitate SQL databases upgrade of a Sitecore XC containerized environment, you also can make use of the new upgrade container to upgrade XC databases stored on-prem, in a non-containerized environment.

You can run the Sitecore upgrade container in Docker Compose or in Kubernetes environments.

> **NOTE**
> If you do not want to use the new Sitecore XC upgrade container to upgrade your XC databases, download and follow the instructions provided in the *Sitecore Experience Commerce Upgrade Guide for Sitecore XC 10.1 to XC 10.2*.

> **WARNING**
> Prior to performing the Sitecore XC upgrade, you must ensure to download and review the release notes and the listed breaking changes.

2. Prerequisites

You can use the Sitecore XC container upgrade procedure to upgrade your Sitecore XC installation if your environment fulfills the following prerequisites:

- You are upgrading a Sitecore XC 10.1 environment to Sitecore XC 10.2
- You have created backup copies of your Sitecore XP and XC databases.

> **IMPORTANT**
> We strongly recommend that you perform the upgrade on copies of your production databases that contain all your data.

- You have created a backup of your Commerce Engine.

2.1. Limitations

This document describes how to upgrade a single Sitecore XC environment. You may need to repeat the process if your Commerce solution is deployed in a distributed environment.

The database upgrade process is irreversible. Before you upgrade a solution, we recommend that you:

- Verify the upgrade process in an isolated environment.
- Perform a backup of your data.

The upgrade process describes how to upgrade Sitecore XC databases located on a single SQL Server instance.

For information about the configuration and upgrade of individual roles and the required post-upgrade steps, see Sitecore Experience Commerce Upgrade Guide for Sitecore XC 10.1 to XC 10.2, available on the Sitecore Downloads site.

2.2. Compatibility

The Sitecore XC upgrade container described in this guide can only be used to upgrade from a Sitecore XC 10.1 version to Sitecore XC 10.2.

See this Knowledge Base article for additional details about Sitecore XC software compatibility for this release.

2.3. Software requirements

Before you proceed with the upgrade, make sure that the environment meets the software requirements.

2.3.1. Software requirements for Docker and for Kubernetes cluster and client

For a list of required software for a Docker host machine or for Kubernetes cluster and client, refer to the Software requirements section in the *Sitecore Upgrade Container Deployment Guide for Sitecore XP 10.2* available on the Sitecore Downloads page.

> **NOTE**
> Since you must upgrade Sitecore Experience Platform before you proceed with the Sitecore XC upgrade process, this guide assumes that when you are ready to start the XC upgrade process, you already have the required Docker or Kubernetes software.

2.3.2. Sitecore XC software requirement

The Sitecore XC upgrade process requires the following Sitecore XC software package available on the Sitecore Downloads site:

- The Sitecore XC Packages for On Premises WDP
- The Sitecore Container SDK (`Sitecore.Commerce.Containers.SDK.10.2.*.zip`) package

3. Overview of the Sitecore XC container upgrade process

The following summarizes the steps you perform to upgrade a Sitecore Experience Commerce 10.1 solution to Sitecore XC 10.2 containerized environment:

- Upgrade Sitecore XP.
- Prepare for Sitecore XC upgrade.
- Deploy the Sitecore XC upgrade container.
- Clean up the environment.
- Build and deploy your custom Commerce solution using containers.
- Perform post-upgrade steps.

4. Upgrade Sitecore Experience Platform

To upgrade Sitecore XP, follow the instructions provided in the *Sitecore Experience Platform Upgrade Container Deployment Guide* for XP 10.2 (available on the Sitecore XP 10.2 Downloads page).

You must complete the following steps as part of the Sitecore XP upgrade before you upgrade your Sitecore Experience Commerce solution:

- Upgrade the Sitecore SQL databases.
- Clean up your environment.
- Upgrade the Sitecore Experience Accelerator (SXA) module in containers (if the environment you are upgrading uses the SXA Storefront).

> **IMPORTANT**
> Sitecore Experience Accelerator is a prerequisite for the SXA Storefront. To upgrade SXA, make sure to follow instructions for XP module upgrades in Sitecore Upgrade Container Deployment Guide, and the SXA upgrade instructions.

5. Prepare for Sitecore XC upgrade

In preparation for the upgrade, you must update some configuration files that are used when you run the upgrade container.

To prepare to upgrade:

- Download and extract the `Sitecore XC Container SDK` package.
- Verify SQL connection strings.
- Update the Docker environment variables .

5.1. Verify SQL connection strings

Verify the definition of the SQL connection strings in the `docker-compose-upgrade.yml` or the `mssql-upgrade.yaml` to make sure that they will be constructed properly for your deployment. If your databases use custom prefix and names, make sure to update these files accordingly so that the connection strings are constructed as expected.

If you do not want to upgrade some of the databases, delete their connection strings from the `docker-compose-upgrade.yml` or the `mssql-upgrade.yaml` file.

By default, the Sitecore XC upgrade container upgrades the following databases:

- Sitecore XC Global DB
- Sitecore XC Shared DB
- Sitecore XC Archive DB
- Sitecore XP Core
- Sitecore XP Master
- Sitecore XP Web

5.2. Update Docker environment variables

The Sitecore.Container.Commerce.SDK contains Docker compose environment variables that are used to pass configuration settings into containers.

> **NOTE**
> If you are upgrading to the Azure Kubernetes Service, skip this step and go to the Deploy the upgrade container to the Azure Kubernetes Service section instead.

The container upgrade process uses these Docker environment variables to resolve the SQL database connection strings listed in the Docker Compose configuration file (`docker-compose-upgrade.yml`).

The Sitecore.Commerce.Container.SDK includes a `ComposeInit` script that automates the preparation steps for running the upgrade container. If you want to use the `ComposeInit` script, skip this step and go to section Deploy the upgrade container in Docker Compose . Otherwise, perform the following step manually to prepare for the upgrade.

To update the environment variables manually:

- In the `Sitecore.Commerce.Container.SDK` folder, open the following environment variables file, and update the values to match those used in your deployment:

  ```
  upgrade/sitecore-xc/upgrade.env
  ```

See Appendix A for a list of the variables included in the `upgrade.env` file.

6. Deploy the Sitecore XC upgrade container

The Sitecore XC upgrade container includes two sets of artifacts that update your Sitecore XC databases, that is, one for Docker Compose, and one for Kubernetes.

This chapter contains the following sections:

6.1. Deploy the upgrade container in Docker Compose

To deploy the Sitecore XC upgrade container in Docker Compose:

1. In Docker for Windows, switch to Windows container mode.
2. In Windows Explorer, go to the folder where you extracted the Sitecore XC Container SDK, and open the folder that contains Docker Compose upgrade artifacts:

   ```
   \upgrade\sitecore-xc.
   ```

3. Open the `upgrade\sitecore-XC\docker-compose-upgrade.yml` file and study its contents for a better understanding of the container and connection strings needed to upgrade the databases
4. To update the environment configuration file with the appropriate values, open a PowerShell window, and run the sample `ComposeInit.ps1` script (located in the `Sitecore.Commerce.Container.SDK/scripts/` folder)

   ```
   /ComposeInit.ps1 -LicenseXmlPath <pathToSitecoreLicense> -SqlServer <SQLServerName> -
   SqlUserName <SqlSaUserName> -SqlSaPassword <SqlSaPassword> -Isolation <IsolationMode>
   ```

   where:

   - `LicenseXmlPath` sets the value of the SITECORE_LICENSE variable.

     NOTE
     You must specify this parameter when you run the `ComposeInit.ps1` script. The script generates the base64 version of the license file that is required, and then adds it the variable file.

   - `SqlServer` sets the value of the SQL_SERVER variable
   - `SqlUserName` sets the value of the SQL_USERNAME variable
   - `SqlSaPassword` sets the value of the SQL_PASSWORD variable

- `Isolation` sets the value of the ISOLATION variable

> **NOTE**
> For more information about running the script to prepare for deployment, see the Sitecore XC Installation Guide for a Developer Workstation with Containers, in the section "Run script to prepare for deployment".

5. In the Windows console, go to the folder that contains the `docker-compose-upgrade.yml` file, and run the following Docker Compose command:

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

The `--env-file` parameter was introduced in Docker Compose 1.25.0.
Docker Compose pulls the `sitecore-xc-mssql-upgrade` image from the Sitecore Container Registry and deploys the container which immediately starts the upgrade process.
When the upgrade process is completed, the upgrade container stops.

6. To check the status of the Docker container, run the following command:

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

This command reports the current status of the upgrade container.

6.2. Deploy the upgrade container to the Azure Kubernetes Service

To deploy the Sitecore XC upgrade container to the Azure Kubernetes Service, you must have a Kubernetes cluster.

To create a new Azure Kubernetes Service (AKS) cluster with a Windows Server 2019 node pool, you can use the Azure command-line interface (Azure CLI) or the Azure portal UI. The AKS cluster must contain one Windows Server 2019 version 1809 node pool with one or more nodes, and the cluster must have access to the Docker registry with the upgrade images.

6.2.1. Configure the kubectl context cluster

Kubectl is the Kubernetes command-line tool that you use to run commands against Kubernetes clusters.

To configure the kubectl context cluster:

1. Log in to the Azure CLI and set a subscription.

```
az login
az account set --subscription <Your subscription>
```

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

```
az aks get-credentials --resource-group <Your resource group> --name <Your cluster name>
```

6.2.2. Update kubernetes.yml and secret files

You use the sample `UpdateK8SUpgradeYaml.ps1` script included in the Sitecore.Commerce.Container.SDK package to update the `kustomization.yml` and Kubernetes secret files.

To update `kubernetes.yml` and secret files:

- Open a PowerShell window, and run the sample `UpdateK8SUpgradeYaml.ps1` script (located in the `Sitecore.Commerce.Container.SDK/scripts/` folder), specifying the appropriate values for your deployment environment, for example:

```
UpdateK8SUpgradeYaml.ps1 -jsonFile 'C:\download\Commerce.Containers\scripts
\configltsc2019.json' `
     -k8sRootPath 'C:\download\Commerce.Containers\upgrade\k8s-sitecore-xc1' `
     -licenseFilePath "Location of your licence file" `
     -sharedDB "Your XC shared database name" `
     -globaldb "Your XC global database name" `
     -archiveDB "Your XC archive database name" `
     -masterKeyPassword "Your XC master key password" `
     -sqlServer "Your sql server instance name" `
     -sqlUsername "Your sql server admin user account name to use" `
     -sqlPassword "The sql server admin account password" `
     -sitecoreDbPrefix "The prefix used for Sitecore databases in your environment" `
     -isAlwaysEncrypted "Is the sql connection always encrypted" `
     -altRegistry "the alternative container registry to use is using private images"
```

6.2.3. Deploy the Sitecore XC upgrade job

To deploy the Sitecore XC upgrade job:

1. Ensure that all secrets files in the configuration folder are updated with correct values.
2. From the root folder run the following command to:
   - Deploy the secrets.
   - Populate the ConfigMap values.
   - Pull the Sitecore upgrade container image from the Sitecore Container Registry.
   - Deploy the Sitecore upgrade job.

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

3. Wait until the status of the job is *Complete/OK.*
4. To check the status of the job with details, run the following command:

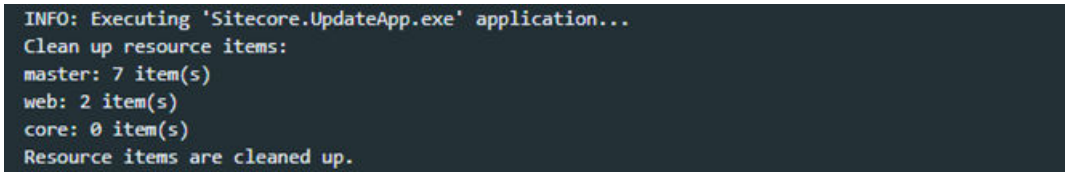
   ```
   kubectl get pods –o wide
   ```

6.3. Verify the status of the upgrade process

The database upgrade process is logged and you can find all the details of the process in the Docker container or in the Kubernetes pod logs.

The logs are saved to the `C:\logs\` folder in the container.

When the databases upgrade process is successfully completed, the following messages appears with `ExitCode 0` in the log and in the terminal:



6.3.1. Troubleshooting

If you encountered any errors related to SQL connections, make sure that the SQL Server is accessible to the upgrade container.

For example, if the following error occurs:

```
Exception:
System.Data.SqlClient.SqlException
Message: A network-related or instance-specific error occurred while establishing a
connection to SQL Server. The server was not found or was not accessible.
Verify that the instance name is correct and that SQL Server is configured to allow remote
connections. (provider: Named Pipes Provider, error: 40 - Could
not open a connection to SQL Server)
```

Perform the following:

- Verify that the SQL server name in the environment file or secrets file is correct.
- Verify that the connection strings being created either in the `docker-compose-upgrade.yml` or the `mssql-upgrade.yml` are constructed to ensure a proper connection.
- Verify that the username and password are correct.

7. Clean up the environment

When the upgrade process is complete, clean up your environment to remove the upgrade container.

7.1. Clean up a Docker Compose environment

To clean up a Docker Compose environment, run the following command:

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

This command deletes the Sitecore XC upgrade container from the Docker system.

7.2. Clean up a Kubernetes environment

To clean up a Kubernetes environment, delete the Kubernetes upgrade job.

To delete the Kubernetes upgrade job:

1. Start a PowerShell session from the Kubernetes upgrade artifacts folder.
2. To delete the mssql upgrade job, run the following cmdlet:

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

3. To delete the upgrade secrets, run the following cmdlet:

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

8. Build and deploy your custom Commerce solution using containers

To build your custom solution and deploy the individual Sitecore roles using Sitecore XC 10.2 containers:

1. Download and use the latest released code base included in the Sitecore XC 10.2 Packages for On Premises WDP.
2. Consume all breaking changes listed in the release notes, and ensure to resolve all conflicts.

   IMPORTANT
   Your custom XC solution must fully align with the new Sitecore Commerce release. For a summary of all policy and configuration changes in Sitecore XC 10.2 that might affect your Commerce Engine customizations, refer to the *Sitecore Upgrade Guide for Sitecore XC 10.1 to XC 10.2*, available for download at this location.

3. Use the new Sitecore Commerce Docker images included in the `Sitecore.Commerce.Container.SDK` package as a base in the Docker files for the corresponding roles, and rebuild your custom Commerce solution images.
4. Deploy the containerized environment using the new Sitecore Commerce roles images with your solution, and configure it to use the upgraded SQL database.

9. Post-upgrade steps

After you have successfully upgraded your Sitecore XC containerized solution, you must complete the following tasks to complete the upgrade:

- Disable antiforgery protection (temporary)
- Bootstrap the Commerce Engine
- Flush the Redis cache
- Synchronize Control Panel content items
- Update the data templates
- Re-enable the anti-forgery setting
- Update the storefront domain configuration
- Rebuild Commerce indexes
- Republish and rebuild the search indexes

9.1. Disable antiforgery protection (temporary)

You must disable antiforgery setting for the Business Tools in the Authoring service configuration file.

> **NOTE**
> When you place a call to the Commerce Engine API from outside the Commerce Business Tools (for example, using Postman), you must disable the antiforgery protection setting in the `wwwroot\config.json` file.

To disable the antiforgery protection:

1. Locate the the anti-forgery setting and change the value to false:

```
"AntiForgeryEnabled": false,
```

9.2. Bootstrap the Commerce Engine

You must bootstrap the Commerce Engine to propagate the changes you made to the environment configuration files into the global database.

The Sitecore XC SDK includes code samples for DevOps operations, so that you can access the Sitecore XC API directly. The following instructions assume that you are using Postman to exercise the Sitecore XC API and that you have imported the Adventure Works and Habitat environments into Postman.

To run the bootstrap operation:

1. Launch the Postman application.
2. In the **Collections** pane in Postman, navigate to the *Authentication* folder.
3. Open the `Sitecore` folder and execute the `GetToken` request.
   When Postman displays an access token in the **Body** pane, authentication is successful.
4. In the **Collections** pane, navigate to the *SitecoreCommerce_DevOps* folder.
5. Open the *1 Environment Bootstrap* folder, and execute the `Bootstrap Sitecore Commerce` request.
6. Restart the container.

9.3. Flush the Redis cache

After you have completed all of the steps required to upgrade the Commerce Engine, you must flush the Redis cache as described in the Redis documentation to remove any content that may have been added to the cache during the upgrade process.

9.4. Synchronize Control Panel content items

As part of the upgrade process, you must synchronize content items in the Control Panel to the Commerce Engine.

If you previously synchronized Sitecore Control Panel content items to the Commerce Engine, you must re-synchronize the content.

If this is the first time Sitecore Control Panel content items are synchronized to the Commerce Engine, use this procedure instead.

Re-synchronize Control Panel content items

To re-synchronized Control Panel content item paths:

1. Launch the Postman application.
2. Navigate to the *ContentAPISamples* folder in the **Collections** pane.
3. Open the *DoActions* folder and execute the `Sync Content Path` sample for each environment that has a different data store (that is, uses a distinct dataset and configuration, identified by a different `ArtifactStoreId`).
   You can use the `Check Long Running Command Status` sample to ensure that the synchronization operation completes, as the operation may take several minutes.

   NOTE
   You may need to obtain the `EntityView` value to use as input for the Sync Content Path sample if you receive an error related the entity version.

4. Launch Internet Information Service (IIS) Manager and perform an IIS restart.

Synchronize Control Panel content items for the first time

If you have not synchronized Control Panel content items before, perform the following steps:

1. Launch the Postman application.
2. Navigate to the *SitecoreCommerce_DevOps* folder in the **Collections** pane.
3. Open the *3 Environment Initialize* folder, and run the `Ensure\Sync default content paths` sample request for each environment that has a different data store (that is, for each environment that use a different `ArtifactStoreId`).

   NOTE
   The operation might take several minutes to complete. You can run the `Check Long Running Command Status` sample in Postman to ensure that the synchronization operation completes.

4. Launch Internet Information Service (IIS) Manager and perform an IIS restart.

9.5. Update the data templates

To update the data templates:

1. Log in to Sitecore Launchpad (`https://<host>/sitecore`) and click **Content Editor**.
2. Click the **Commerce** tab.
3. Click **Delete Data Templates**.
4. Click **Update Data Templates**.

9.6. Re-enable the anti-forgery setting

Re-enable the anti-forgery setting for the Business Tools in the Authoring service configuration file.

To enable the anti-forgery setting:

- Locate the the anti-forgery setting and change the value to true:

```
"AntiForgeryEnabled": true,
```

9.7. Update the storefront domain configuration

If there is an existing storefront site in your current Sitecore XC deployment, update the configuration in the new `domain.config` file with your own storefront domain configuration. The location of this configuration is different for containerized deployment.

9.7.1. Update the storefront domain in a non-containerized deployment

To update the storefront domain configuration in a non-containerized deployment:

- Open the new `\inetpub\wwwroot\<sitename>\app_config\security\domains.config` file, and update the configuration to reflect the domain name of your storefront site.

9.7.2. Update the storefront domain in a containerized deployment

In a container deployment, the `domains.config` is stored in a shared volume.

To update the storefront domain configuration for your site in containers:

- In Docker, open the `C:\containers\<cd or cm>\domains-shared\domains.config` file, and update the domain information as required.

9.8. Rebuild Commerce indexes

The schema update to Solr 8.8.2 and the introduction of new indexes require a complete rebuild of search indexes.

To rebuild the Sitecore XC indexes:

> **NOTE**
> Before you rebuild the Sitecore indexes, make sure that previous indexes were deleted.

1. Launch the Postman application
2. In the **Collections** pane, navigate to the *SitecoreCommerce_DevOps* folder.
3. Expand the *Minions* folder and execute the following requests:

   > **NOTE**
   > When you execute the indexing minion requests, make sure that they point to the URL of the Commerce Engine instance that is running the Minions service.

   - `Run FullIndex Minion - Orders`
   - `Run FullIndex Minion - Customers`
   - `Run FullIndex Minion - PriceCards`
   - `Run FullIndex Minion - Promotions`
   - `Run FullIndex Minion - Catalog Items`

9.9. Republish and rebuild the search indexes

To republish the site and re-index the core, master and web indexes:

1. Log in to Sitecore Launchpad (`https://<host>/sitecore`) and click on **Content Editor**.
2. Click the **Publish** tab.
3. Click **Publish** and select **Publish site** from the drop-down menu.
4. In the **Publish Site** window, select **Republish – publish everything** and click **Publish**.
5. Return to the Sitecore Launchpad and click on **Control Panel**.
6. Click on **Indexing manager**.
7. Select the following indexes from the Local Indexes list and click **Rebuild**:
    - `sitecore_core_index`
    - `sitecore_master_index`
    - `sitecore_web_index`

10. Appendix A - Docker environment variables for upgrade

The following tables lists the environment variables included in the `update.env` file:

Variable name	Default value	Description
BASE_SITECORE_DOCKER_REGISTR	src.sitecore.com/base/	The base Sitecore container registry.
XP_SITECORE_DOCKER_REGISTRY	scr.sitecore.com/sxp/	Sitecore container registry.
XP_SITECORE_TAG	10.2-ltsc2019	Sitecore XP image tag.
XC_NONPRODUCTION_SITECORE_DOCKER_REGISTRY	scr.sitecore.com/sxc/nonproduction/	The Sitecore Commerce container registry for non-production container.
XC_SITECORE_DOCKER_REGISTRY	scr.sitecore.com/sxc/	Sitecore Commerce container registry.
XC_PACKAGES_TAG	10.2-ltsc2019	Short tag for Sitecore XC containers.
ISOLATION	default	Override for Docker isolation modes.
XC_GLOBAL_DB	SitecoreCommerce_Global	The Sitecore Commerce global database name.
XC_SHARED_DB	SitecoreCommerce_SharedEnvironments	The Sitecore Commerce shared database name.
XC_SHARED_ARCHIVE_DB	SitecoreCommerce_ArchiveSharedEnvironments	The name of the database where archived Commerce entities are stored.
SQL_SERVER	*no default*	The name of the MS SQL server.
SQL_USERNAME	*no default*	The user name for accessing the SQL server.
SQL_PASSWORD	*no default*	The password for the SQL user account.
IS_ALWAYS_ENCRYPTED	false	Specifies whether the SQL Always Encrypted feature is enabled.
SQL_DATABASE_PREFIX	Sitecore	The prefix to use in SQL database names.
SITECORE_LICENSE	*no default*	The path to the Sitecore license file, in a Base64 encoded format.

NOTE
The `Init Compose.ps1` generates the required base64 version of the license file, and sets the variable in the `update.env`file.