Sitecore 7.5

xDB™ Upgrade and Conversion Guide

Step by step guide to the Sitecore 7.5 upgrade and conversion process.

Table of Contents

Chapter 1

Introduction

The Sitecore® Experience Database™ (xDB™) collects online and offline customer interactions from all channel sources in a real-time big data repository. It connects interaction data to create a comprehensive, unified view of each individual customer, and makes the data available to marketers to manage the customer experience in real time.

If you are running an earlier version of Sitecore and want to upgrade to the Sitecore Experience Platform, follow the steps outlined in this document.

A system administrator or developer should perform these tasks.

This chapter contains the following sections:

- Overview
- Glossary of Terms

1.1 Overview

This document covers the steps you need to follow for both the upgrade and conversion process.

Upgrade

Follow the steps outlined in the upgrade section if you want to upgrade from an earlier version of Sitecore to be compatible with the Sitecore Experience Platform.

There are several reasons why you might want to upgrade.

The main advantages are:

- Scalability and performance – you can distribute your installation horizontally across multiple servers in multiple geographical locations, adding more servers over time to expand your solution without significantly affecting performance.
- Availability and storage – all data is stored and made available to the latest Sitecore reporting applications.

For more detail on the specific steps you need to follow see, Chapter 2 *Upgrade Process*.

Optimization

Follow the steps outlined in the Data Optimization chapter in order to analyze and fix data from an earlier version of Sitecore. To do this you need to use the Sitecore Analytics Data Optimization Tool.

For more detail on the specific steps you need to follow to use the tool see, Chapter 3 *Data Optimization*.

Conversion

Follow the steps outlined in the Database Conversion chapter if you want to import and convert analytics data from an earlier version of Sitecore to the Sitecore Experience Platform. To do this you need to use the Sitecore Database Conversion Tool.

Summary of steps in the conversion process:

- Run the Data Optimization Tool (prepares and optimizes the data contained in the Analytics database).
- Run the Sitecore Database Conversion Tool.
- Synchronize the reporting database with the collection database (rebuild the reporting database).

For more detail on the steps you need to follow to convert data from an earlier Sitecore installation see, Chapter 3 Database Conversion.

Rebuilding Reporting Database

Follow the steps outlined in the chapter Rebuilding the Reporting Database in order to rebuild reporting database.

For more detail on the steps you need to follow to convert data from an earlier Sitecore installation see, Chapter 5 *Rebuilding the Reporting Database*.

1.2 Glossary of Terms

This section contains some key terms used in the upgrade and conversion process. For more Sitecore Experience Database terms and definitions, see the *xDB Overview and Architecture* document.

Term	Definition
Collection database	The collection database (MongoDB) is a NoSQL database that stores contacts, history, and automations and is capable of storing vast amounts of customer data.
NoSQL database	The collection database is a NoSQL database. A NoSQL database is non-relational database, such as MongoDB.
Reporting database	The reporting SQL Server database is a cache for pre-aggregated analytics reporting data. This cache is used by Sitecore reporting applications such as the *Executive Insight Dashboard*, *Engagement Analytics*, and *Engagement Intelligence*.
Processing	Describes all processing performed in the processing layer. For example, aggregation, live processing, and history processing.
Aggregation	Aggregation, the process of grouping and reducing data from the collection database.
Aggregator	The aggregator is the data processor used during aggregation.
Live processing	One type of live processing is aggregation. This groups and reduces data from the collection database and stores it in the reporting SQL Server database for use by reporting applications.
History processing	History processing requires you to rebuild the reporting database. Perform this step after you have re-classified a search key word or traffic type. All existing data contained in the reporting database is overwritten every time you perform a rebuild.

Term	Definition
Processing pool	When a visitor session ends, it is saved to the collection database and a unique identifier for the visit is added to the processing pool. This notifies the aggregation framework that there is a new visit ready to be processed. The processing pool is a collection in a MongoDB database.
History worker	History workers are processes that manage the task of reprocessing history data stored in the collection database.

Chapter 2

Upgrade Process

If you have an earlier version of Sitecore installed and you want to upgrade to Sitecore 7.5 then follow the instructions in each section of this chapter.

As part of the upgrade process, some solutions might require redesigning, so if you need more help and guidance, refer to the administrator and developer documentation on the Sitecore Developers Network (SDN).

This chapter contains the following sections:

- Prerequisites
- Installing the Upgrade Package
- Updating Configuration Files
- Rebuilding Search Indexes

2.1 Prerequisites

If your solution is based on an earlier version of Sitecore, then you must first update to Sitecore 7.2 rev. 140228 or later before you can install the Sitecore 7.5 upgrade package. You can download all prerequisite components from SDN.

Prerequisites for running this update:

1. Sitecore CMS 7.2 rev. 140228 or later.
2. Sitecore CMS 7.5 update package.
3. Execute the SQL Server script `UpgradeCMS75_sql.zip` file before you install the upgrade package.
4. Two clean SQL Server Sitecore 7.5 reporting databases (one primary and one secondary).
5. Configuration files

   In Sitecore 7.5, you may need to configure dedicated server roles for content management, content delivery, processing servers, and MongoDB. The configuration of these server roles requires the use of several different configuration files.

Note
Upgrading a solution that uses Oracle databases is currently not supported.

2.2 Installing the Upgrade Package

To install the upgrade package to update you website from Sitecore CMS 7.2 to 7.5 you need to use the update installation wizard.

Before you install the upgrade package:

1. Backup your website.
2. Make your Sitecore databases compatible with bacpac:

   Execute the `UpgradeCMS75_sql.zip` SQL Server script on each of your Sitecore core, master, and web databases.
3. To avoid interrupting the upgrade process, disable the following configuration files by adding `.disabled` to the file extension:

   - `Sitecore.Analytics.config`
   - `Sitecore.Analytics.ExcludeRobots.config`
   - `Sitecore.Speak.config`

   If you have also installed modules, disable the following files:

   - `Sitecore.EmailCampaign.config`
   - `Forms.config`

   Remember to enable these files again when the upgrade process is complete.

   Note
   After you have installed the upgrade package but before you update your modules, add the Analytics include files to the `App_Config\Include` folder that come with the latest version of the Sitecore xDB. For more information, see the section *Updating Configuration Files*.
4. Install MongoDB.

   For more information on installing MongoDB, see the *xDB Configuration Guide* on SDN and the MongoDB documentation on www.mongodb.org.
5. In the `App_Config/ConnectionStrings.config` file, configure connection strings to the collection database (MongoDB) and the reporting database (SQL Server).

   For more detailed information, see the section *Updating Configuration Files* (step 3).
6. If you are upgrading from Sitecore 7.2 rev. 150408 Update-4, in the `Sitecore.ContentSearch.config` file, in the `<events>` section, remove the following events:

```
      <event name="publish:end:remote">
        <handler type="Sitecore.ContentSearch.Events.PublishingEventHandler,
Sitecore.ContentSearch" method="OnFullPublishEndRemoteHandler"/>
      </event>
      <event name="publish:end">
        <handler type="Sitecore.ContentSearch.Events.PublishingEventHandler,
Sitecore.ContentSearch" method="OnFullPublishEndHandler"/>
      </event>
      <event name="publish:end">
        <handler type="Sitecore.ContentSearch.Events.PublishingEventHandler,
Sitecore.ContentSearch" method="OnPublishHandler"/>
      </event>
      <event name="publish:end:remote">
        <handler type="Sitecore.ContentSearch.Events.PublishingEventHandler,
Sitecore.ContentSearch" method="OnPublishRemoteHandler"/>
```

```
      </event>
      <event name="indexing:updateditem">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampHandler"/>
      </event>
      <event name="indexing:deleteitem">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampHandler"/>
      </event>
      <event name="indexing:deletegroup">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampHandler"/>
      </event>
      <event name="indexing:end">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampDirectHandler"/>
      </event>
      <event name="indexing:committed">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampDirectHandler"/>
      </event>
      <event name="indexing:end:remote">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="IndexEndedRemoteHandler"/>
      </event>
      <event name="indexing:end:remote">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="UpdateIndexTimestampDirectHandler"/>
      </event>
      <event name="packageinstall:starting">
        <handler type="Sitecore.ContentSearch.Events.PackagingEventHandler,
Sitecore.ContentSearch" method="OnPackageInstallStartingHandler"/>
      </event>
      <event name="packageinstall:poststep:starting">
        <handler type="Sitecore.ContentSearch.Events.PackagingEventHandler,
Sitecore.ContentSearch" method="OnPackagePostStepInstallStartingHandler"/>
      </event>
      <event name="packageinstall:items:ended">
        <handler type="Sitecore.ContentSearch.Events.PackagingEventHandler,
Sitecore.ContentSearch" method="OnPackageInstallItemsEndHandler"/>
      </event>
      <event name="packageinstall:ended">
        <handler type="Sitecore.ContentSearch.Events.PackagingEventHandler,
Sitecore.ContentSearch" method="OnPackageInstallerEndHandler"/>
      </event>
```

7. If you are upgrading from Sitecore 7.2 rev. 150408 Update-4 or later, in the `Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config` file, remove the following elements:

   In the `<fieldNames hint="raw:AddFieldByFieldName">` section, remove:

```
<field fieldName="culture" storageType="YES" indexType="TOKENIZED" vectorType="NO"
boost="1f" type="System.String"
settingType="Sitecore.ContentSearch.LuceneProvider.LuceneSearchFieldConfiguration,
Sitecore.ContentSearch.LuceneProvider">
```

   and

```
<analyzer type="Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer,
Sitecore.ContentSearch.LuceneProvider" />
</field>
```

In the `<fields hint="raw:AddComputedIndexField">` section, remove:

```
<field
fieldName="culture">Sitecore.ContentSearch.ComputedFields.Culture,Sitecore.ContentSearch<
/field>
```

8. If you are upgrading from Sitecore 7.2 rev. 151021 Update-5 or later, in the `web.config` file, comment out the following line:

```
        <!--  <add type="Sitecore.Web.HttpModuleDisabler, Sitecore.Kernel"
 name="SitecoreHttpModuleDisabler" /> -->”
```

9. If you are upgrading from Sitecore 7.2 rev. 151021 Update-5 or later, in the `Sitecore.ContentSearch.config` file, in the `<events>` section, remove the following events:

```
      <event name="indexing:start">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="IndexingStartedHandler" />
      </event>
      <event name="indexing:end">
        <handler type="Sitecore.ContentSearch.Events.IndexingEventHandler,
Sitecore.ContentSearch" method="IndexingEndedHandler" />
      </event>
```

10. Make sure that the IIS is configured to allow access to the `/sitecore/admin` folder.

Note
This step may be necessary if you install Sitecore using the `setup.exe`. The `setup.exe` configures the IIS to disallow access to the `/sitecore/admin` folder and prevents you from using the *Update Installation Wizard*.

To install the upgrade package:

1. Download the latest `Newtonsof.Json.dll` from the SDN or directly from the latest Sitecore 7.5 installation package available, and copy it to the `website/bin` folder of the Sitecore website that you want to upgrade.
2. Open the `Web.config` file, navigate to the `<dependentAssembly>` section, and update the `Newtonsoft.Json bindingRedirect` to redirect to the latest version.
3. Open the *Update Installation Wizard* by entering the following URL in your web browser:

   `http://<hostname>/sitecore/admin/UpdateInstallationWizard.aspx`

   See the *Update Installation Wizard* guide on SDN for more information on how to use the wizard.
4. Install the Sitecore 7.5 update package using the *Update Installation Wizard*.

   Summary of steps:

   a. Upload the update package
   b. Analyze the package
   c. Install the package

Note
“*Item not found*” collisions are expected for translation dictionary items. It is recommended to apply the latest translation to your instance after upgrading to keep it up to date.

For more detailed information, see Warnings and Collisions at the end of this section.

5. Enable the following file: `Sitecore.Speak.config`

Note
Only enable the Sitecore.Speak.config file if you had SPEAK enabled in your 7.2 solution.

6. Attach an empty reporting databases in SQL Server (MSSQL database by default). Make this the primary reporting database and name it `reporting` in the connection string. You can use the `Sitecore.Analytics.mdf` SQL Server database that comes with the DMS 7.5 package.
7. Attach one more, empty reporting database. Make this the secondary reporting database and name it `reporting.secondary` in the connection strings file.

Note
The reporting secondary database is used as part of the database conversion process when you transfer live and historical data by rebuilding the reporting database.

Note
If you are using the Microsoft `StateServer` session state provider and you are also using Analytics, then you should switch to using the *Sitecore ASP.NET Session State Store Provider for Microsoft SQL Server*. The Microsoft `StateServer` provider does not support the `SessionEnd` event which is required for Analytics.

2.2.1 Warnings and Collisions

The following warnings and collisions are expected.

Message	May apply to the following items or their sub-items:
"*Item has been moved*"	• `/sitecore/system/Settings/Analytics/Reports SQL Queries` • `/sitecore/templates/System/Analytics/ReportQuery/Queries`
"*Item to be deleted is in use*"	• `/sitecore/system/Settings/Rules/Definitions/Obsolete/Marketing Workflows/Conditions/Automation State/Is Mail Opened` • `/sitecore/system/Settings/Rules/Definitions/Obsolete/Marketing Workflows/Conditions/Automation State/Is Mail Clicked` • `/sitecore/system/Settings/Rules/Definitions/Elements/Visit/Send Notifications`
"File has been modified"	• `Website\bin\Newtonsoft.Json.xml`
"Item not found"	• `/sitecore/system/Dictionary/F/From_2`

Note
If you upgrade your solution from Sitecore CMS 7.2 Update-1 or Update-2, the number of collisions and warnings might be higher because CMS 7.5 upgrade package is cumulative.

2.3 Updating Configuration Files

After installing the update package you need to update your configuration files.

Due to the significant number of changes in the configuration we recommend that you perform the following steps to complete the upgrade process:

1. Remove any configuration files which you have changed to customize your analytics settings, as these may now be obsolete.
2. Copy the default `Sitecore.Analytics.config` configuration file into the `App_Config/Include` folder along with all the other Analytics configuration files that come with your Sitecore CMS installation. Overwrite any existing files.

   Note
   Before overwriting files first make a note of any settings you have changed or any customizations that you have made directly to the file.

   Ensure that you carry out this step after you have completed the upgrade process but before you upgrade your modules.

3. Edit the default `ConnectionStrings.config` file and save it to the `App_Config` folder. Update the settings in this file according to your environment. For example, at minimum you need to configure connection strings to the collection database (MongoDB) and the reporting database (SQL Server).

   For more information on configuring connection strings, see the *xDB Configuration Guide* on SDN.

4. Update your configuration files:
   - Apply the configuration file changes described here:

     Configuration file changes in CMS and DMS 7.5 rev. 141003
   - If you are updating to the current major version from 7.2 rev. 140314 (Update-1) or earlier, you must also apply these configuration file changes:

     Configuration file changes in CMS and DMS 7.2 rev. 140526
   - If you are updating to the current major version from 7.2 rev. 140228 (Initial Release), you must also apply these configuration file changes:

     Configuration file changes in CMS and DMS 7.2 rev. 140314
5. Update the analytics configuration according to the requirements of your solution. These requirements are specific to each installation and could be different for every customer. For example, you may have added some configuration settings related to the *Email Campaign Manager* module settings or other related custom functionality in your solution.

   Note
   If you have your own customizations or references based on previous analytics assemblies, then you may need to rebuild your project. This is due to the high number of refactoring changes and improvements made for the Sitecore Experience Platform.

6. To make your Sitecore solution more secure:
   - Open the `App_Config/Include/Sitecore.Media.RequestProtection.config` file and change the `Media.RequestProtection.SharedSecret` setting to a random string.

- If you have multiple Sitecore instances, use the same value for `Media.RequestProtection.SharedSecret` on every instance. Otherwise, dynamic image scaling will not work correctly when the image URL is generated by one instance and the request is handled by a different instance.

7. Upgrade any modules you have installed to make them compatible with the current latest version of Sitecore and you may also need to enable the following configuration files:
   - `Sitecore.EmailCampaign.config`
   - `Forms.config`.
8. If you need to transfer and convert collected data into the new format, follow the instructions in Chapter 3 *Database Conversion*.
9. If you use the Solr provider, perform the following steps:
   - Extract the `Sitecore.ContentSearch.Solr.Indexes.Analytics.config` file from the `Sitecore.Solr.Support 1.0.0 rev. 141001.zip` archive and add it to the `App_Config\Include` folder.
   - Generate a Solr schema using the Solr Schema Builder.
   1) In the **Sitecore Control Panel**, click **Indexing** and then open the **Generate the Solr Schema.xml file** dialog box.
   2) Specify a source and target file and then click **Generate**.
   - Manually remove all old indexes from the Solr core. Solr keeps redundant fields in indexes which during rebuild might cause an exception relative to the *Solr Segment Merge*.

     Note
     The default path to the Solr core is: `SolrServer\example\solr\{Solr core for Sitecore instance}`

   - Restart Solr.

After you change the configuration files, you must clear your browser cache.

To do this:

1. Open Internet Explorer, click **Tools**, **Internet Options**.
2. In the **Internet** Options dialog box, in **Browsing History**, click **Delete**, and then delete all your **Temporary Internet Files**.

These steps may vary depending on the type of browser you are using.

Note
If you temporarily disabled any configuration files to avoid interrupting the upgrade process, remember to enable them again when the upgrade process is complete.

2.4 Rebuilding Search Indexes and the Link Database

To complete the upgrade process you need to rebuild your search indexes.

1. Rebuild all `ContentSearch` indexes.

   Steps: In the Sitecore Desktop, **Control Panel**, click **Indexing**, then open the **Indexing Manager** dialog box. Select all indexes and click **Rebuild**.

2. Rebuild your search indexes for the **Quick search index** (also for the Master and Core databases, if you have them).

   Steps: In the Sitecore Desktop, **Control Panel**, click **Database**, open the **Rebuild the Search Index** dialog box. Select the **Quick search index** and click **Rebuild**.

3. Rebuild the link database for the master and the core databases.

   Steps: In the Sitecore Desktop, **Control Panel**, click **Database**, open the **Rebuild the Link Database** dialog box. Select the core and master databases and then click **Rebuild**.

2.5 Upgrading Multiple Instances

Repeat all the steps contained in the Upgrade Process (this chapter) for every Sitecore instance in your environment that you want to update.

When you have completed the upgrade you can move on to the Data Optimization process which is necessary before you start Database Conversion.

Chapter 3

Data Optimization

If you have an earlier version of Sitecore installed that you want to upgrade then you need to use the Sitecore Analytics Data Optimization Tool to analyze and fix errors in your analytics database before converting your data to be compatible with Sitecore xDB.

- Preparing for Data Optimization
- Running the Data Optimization Tool
- Handling Data Optimization Errors

3.1 Preparing for Data Optimization

To perform data optimization you will need the Sitecore Analytics Data Optimization Tool (command line tool) to analyze and fix errors in the database that you want to convert.

1. Download the Data Optimization Tool.zip package from the same page as the latest version of Sitecore 7.5 on SDN.
2. Unzip the Sitecore Analytics Data Optimization Tool.zip package to a local folder on your computer.

3.1.1 Creating Indexes

An important part of preparation for the conversion process is to create indexes in your Sitecore 7.2 SQL server database that is going to be converted to Sitecore 7.5. Creating indexes can help to speed up searching in the database and also increase data conversion speed.

To create indexes you need to:

1. Open Microsoft SQL Server Management Studio and create indexes in the Sitecore DMS SQL Server database that you want to convert.
2. To do this, right click the Sitecore DMS SQL Server database and then click **New Query**.
3. In the new query tab window execute the following SQL Server script:

```
CREATE NONCLUSTERED INDEX [IDX_ExternalUser] ON [dbo].[Visitors] ([ExternalUser]);
CREATE INDEX [IX_LocationId] ON [NotificationSubscriptions] ([LocationId]);
CREATE NONCLUSTERED INDEX [IDX_Language] ON [dbo].[Visits] ([Language]);
```

3.2 Running the Data Optimization Tool

The Data Optimization Tool prepares, optimizes, and fixes analytics data stored in a Sitecore DMS SQL Server database. You need to run the Sitecore Analytics Data Optimization Tool before you upgrade Sitecore 7.2 analytics data to be compatible with the xDB collection database (MongoDB).

Summary of Steps to follow

Summary of the steps to follow when using the Data Optimization Tool:

1. Run the Data Optimization Tool without the `-fix` flag – this is analysis stage.
2. If you see any issues with indexes, the tool stops, you create indexes then run the tool again without the `-fix` flag.
3. If you get a `CultureInfo` error then update the `LanguagesMap` section in the `DataOptimizationTool.exe.config` and run the tool again without the `-fix` flag.
4. This time you should not get any errors.
5. If you do not get any errors then run tool again, this time with the `-fix` flag.

The Data Optimization Tool automatically fixes all issues it finds in the analytics database (except languages). For more information on how to fix errors, see the section *Handling Data Optimization Errors*.

Running the Data Optimization Tool

To run the data optimization tool:

1. Open a command prompt window or Windows PowerShell.
2. In the command prompt, enter a command that includes the SQL Server connection string parameter to the analytics database that you want to analyze.

   Required parameter format using SQL Server authentication:

   ```
   <PATH_TO_THE_TOOL>\DataOptimizationTool.exe "user id=<USER>; password=<PASSWORD>; Data Source=<SERVER_NAME>; Database=<DB_NAME>”
   ```

   Example:

   ```
   C:\dmsdot\DataOptimizationTool.exe "user id=sa; password=MyPassword; Data Source=.; Database=My_DMS20_DB”
   ```

3. While the tool is running you can see the progress of the analysis:

4. If the Data Optimization Tool has finished without errors you can move on to the database conversion process. If you get errors, see the next section for some possible solutions.

3.3 Handling Data Optimization Errors

During the running of the Data Optimization Tool you may get errors. This section includes guidance on how to handle the following errors.

3.3.1 Missing Index Errors

If the Data Optimization Tool cannot find a SQL Server search index for a specific language you get the following error: `Index for Visits. Language does not exist.`

If you get this error it means you have not created a SQL Server search index for a particular language. The solution to this problem is to execute the following SQL Server script before running the data optimization tool.

```
CREATE NONCLUSTERED INDEX [IDX_Language] ON [dbo].[Visits] ([Language]);
```

Check that you have executed all required scripts for creating search indexes correctly and then run the tool again.

For more information on running SQL Server search index scripts, see the section *Creating Indexes*.

3.3.2 Culture Info Errors

If the Data Optimization Tool finds visit data that contains culture names or information that is not recognized by the .Net `CultureInfo` class you may get a culture info error.

For example, if the language column of the Visit table contains languages different from those registered in .Net `CultureInfo` class this could lead to errors in your data that could make it difficult to aggregate analytics data during the history reprocessing stage.

You may also encounter problems if you are analyzing data that contains a culture setting that is not recognized by.Net `CultureInfo` or if it is not registered as a custom setting.

When you run the data optimization tool it will notify you of errors or any potential problems. To solve this issue you can create language mappings between unrecognized cultures and cultures you want to recognize.

Read the follow instructions to handle culture info errors:

1. No action is necessary if the custom culture is registered on a server but not on your local computer.
2. To convert data that is not recognized by the .Net `CultureInfo` class to be compatible with the `CultureInfo` class, you must update the `DataOptimizationTool.exe.config` file to map the custom culture names to the standard .Net `CultureInfo` names.

Example

```
<add name='[N/A]' value='en-US'/> OR <add name='DK'      value='da-DK'/>
```

Configure your mappings in the language map section of the `DataOptimizationTool.exe.config` file as shown in the following example:

```
<LanguagesMap>
    <Languages>
        <add name="[N/A]" value="en"/>
        <add name="" value="en"/>
    </Languages>
</LanguagesMap>
```

3.3.3 Fixing Culture Info Errors

After you have created the mappings you need, run the Data Optimization Tool again to fix the errors. Languages will be updated to use the new mappings.

1. To run Data Optimization Tool for fixing of DMS 2.0 Analytics database errors add the following `–fix` flag to the command:

```
<PATH_TO_THE_TOOL>\dmsdot.exe "user id=<USER>; password=<PASSWORD>; Data
Source=<SERVER_NAME>; Database=<DB_NAME>” –fix
```

2. While the tool is running and fixing errors you get some feedback on the progress of the correction progress:

Chapter 4

Database Conversion

Follow the steps in this chapter to convert historical data from an earlier version of Sitecore to be compatible with the Sitecore Experience Platform.

This chapter contains the following sections:

- Preparing for Conversion
- Running the Conversion Tool
- Messages and Notifications

4.1 Preparing for Conversion

When you upgrade to the latest version of Sitecore you also need to use the Sitecore Analytics Conversion Tool (command line tool) to convert your historical analytics data.

The conversion tool copies and merges historical analytics data from a Sitecore DMS SQL Server database to the MongoDB collection database.

If you want to ensure that you do not lose any live data during the conversion of historical data you should consider the following approach:

1. Prepare a solution based on Sitecore xDB.
2. First run it on a staging server and prepare for conversion.
3. Swap to a production server (system starts gathering of live data).
4. Start the conversion process on the production server.

Result - all data collected will be registered in the new xDB solution.

4.1.1 System Requirements

The Sitecore Database Conversion Tool uses Sitecore 7.5 and DMS 2.0 API – The system requirements are the same as for Sitecore 7.5. We recommend that you run one instance of the conversion tool per computer and set 20 to 30 threads per instance.

4.1.2 Installation

1. Download the Sitecore Analytics Conversion Tool zip file from the same page as the latest version of Sitecore 7.5 on SDN.
2. Unzip the Sitecore Analytics Conversion Tool to a local folder on your computer.
3. Copy your Sitecore `license.xml` file to the same folder as the conversion tool. If you forget to do this the tool will throw the following exception:

   ```
   Sitecore.SecurityModel.License.LicenseException: License is not valid:
   Not Present.
   ```

   You can now open a command prompt window and start using the tool. For more information, see the section *Running the Conversion Tool*.

   Note
   The Sitecore Analytics Conversion Tool uses a copy of the same license file as Sitecore CMS.

4.1.3 Connection String Parameters

When you run the conversion tool you need to add the correct connection string parameters using the command line. This section contains guidance on the correct connection parameters to use for conversion.

You need to enter two connection string parameters which enable the conversion tool to connect to:

- SQL Server analytics database - historical analytics data
- MongoDB collection database - live and converted data

The full command including connection parameters is written in the following format:

```
<PATH TO CONVERSION TOOL>\ConversionTool.bat /sqlconnection "<DMS2.0 connection string>"
/xdbconnection "xDB connection string" [optional_parameters]
```

Example 1 - Analytics database connection string parameters

Enter connection string parameters to a Sitecore DMS SQL Server database that contains your historical analytics data.

Required parameter format using Windows authentication:

```
"integrated Security=SSPI; Data Source=<SERVER_NAME>; Database=<DB_NAME>"
```

Example:

```
"integrated Security=SSPI; Data Source=.; Database=My_DMS20_DB"
```

Required parameter using SQL Server authentication:

```
"user id=<USER>; password=<PASSWORD>; Data Source=<SERVER_NAME>; Database=<DB_NAME>"
```

Example 2 - Collection database connection string parameters

Enter connection string parameters to a MongoDB collection database, which contains live and converted data.

Required parameter format:

```
"mongodb://<SERVER_NAME>/<DB_NAME>"
```

Example:

```
"mongodb://localhost/My_xDB_DB"
```

4.1.4 Optional Parameters

Currently the Sitecore Analytics Conversion Tool has one important optional parameter that you can use to fine-tune the conversion process. You can set the number of threads that you want to run simultaneously.

When you run the tool, add the following parameter:

```
/threads <N>
```

This parameter defines the maximum number of threads allowed to run in parallel by one instance of the Sitecore Analytics Conversion Tool.

- Default value is 1.
- Recommended value is 20-50.

4.1.5 Configuring the Conversion Tool Across Multiple Servers

If there are not enough resources on a single computer to complete the conversion process in a reasonable time or if you want to increase conversion speed, you can run the Sitecore Analytics Conversion Tool concurrently on multiple computers, provided they all connect to the same two databases. When new instances start, they reschedule the work dynamically.

Note
If there is no improvement in performance by running the conversion tool across multiple servers you can choose to stop the process at any time by closing the application.

If you follow this approach then we recommend you install SQL Server and MongoDB on separate server instances servers and run several instances of the Sitecore Analytics Conversion Tool. If you run multiple instances of the conversion tool you should ideally run them on different computers.

However, if CPU, network, and RAM are not loaded to more than 50% then you can run additional instances of the Sitecore Analytics Conversion Tool concurrently on the same computer.

4.1.6 Conversion Plugins for the ECM and WFFM Modules

If you have installed the Email Campaign Manager or the Web Forms for Marketers module, you must install the module conversion plugins for the Sitecore Database Conversion Tool before the converting the databases.

The database conversion tool loads all the plugins automatically.

Installing the ECM Conversion Plugin

To install the ECM conversion plugin:

1. Download the ECM conversion plugin from the ECM Module Upgrade page.
2. Extract the files from the archive to the `plugins` directory that is located in the same directory as the `Sitecore.Analytics.ConversionTool.exe` file. The plugin contains the following files:
   - `Sitecore.EmailCampaign.Analytics.ConversionPlugin.dll`
   - `Sitecore.EmailCampaign.Analytics.Model.dll`

Installing the WFFM Conversion Plugin

To install the WFFM conversion plugin:

1. Download the WFFM conversion plugin from the WFFM Module Upgrade page.
2. Extract the files from the archive to the `plugins` directory that is located in the same directory as the `Sitecore.Analytics.ConversionTool.exe` file. The plugin contains the following files:
   - `MSCaptcha.dll`
   - `Sitecore.Forms.ConversionTool.dll`
   - `Sitecore.Forms.Core.dll`
   - `System.Data.SQLite.dll`
3. Specify the following required command line parameters for the plugin:
   - `wffmtarget` – the connection string to the target database for the WFFM converted data. If the parameter value is not specified, the conversion tool uses the value of the `xdbconnection` parameter instead.

     Example value:
     ```
     "mongodb://localhost/wffm_analytics"
     ```
   - `wffmsource` – the connection string to the source WFFM database.

     An example of a SQL server connection string:
     ```
     "Data Source=.;Database=Conv_WebForms;Integrated Security=True"
     ```

     An example of a SQLLite connection string:
     ```
     "D:\sc\wffm\Website\Data\Sitecore_WebForms.db;version=3;BinaryGUID=true"
     ```

- `wffmreadertype` – the data reader type. The available readers are:
  - SQL server reader:

```
"Sitecore.Forms.ConversionTool.Reader.SqlServerReader,
Sitecore.Forms.ConversionTool"
```

  - SQLLite reader:

```
"Sitecore.Forms.ConversionTool.Reader.SqlLiteReader,
Sitecore.Forms.ConversionTool"
```

Note
If Analytics is disabled for web forms on a website, the submitted data will not be converted.

4.2 Running the Conversion Tool

This section includes step-by-step instructions on how to run the conversion tool for the first time.

To run the conversion tool:

1. Open a command prompt window or Windows PowerShell.
2. In the command prompt enter the following command. Ensure that you enter the correct connection string parameters:

```
<PATH_TO_CONVERSION_TOOL>\ConversionTool.bat /sqlconnection "<DMS2.0 connection string>"
/xdbconnection "xDB connection string" [optional_parameters]
```

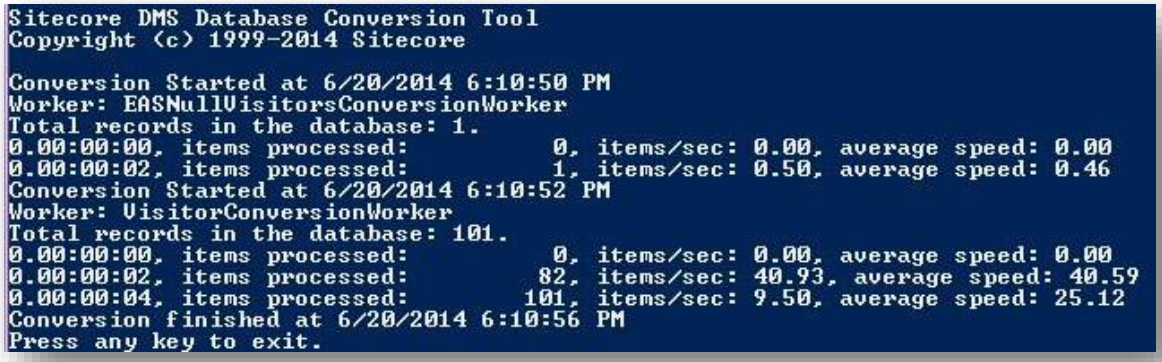
Example:

```
C:\ConversionTool.bat /sqlconnection "user id=sa; password=MyPassword; Data Source=.;
Database=My_DMS20_DB" /xdbconnection "mongodb://localhost/My_xDB_DB" /threads 20
```

Example of running the conversion tool with the WFFM plugin:

```
C:\ConversionTool.bat /sqlconnection "user id=sa; password=MyPassword; Data
Source=.;Database=My DMS20 DB" /xdbconnection "mongodb://localhost/My xDB DB" /threads 20
/wffmtarget "mongodb://localhost/My_xDB_DB" /wffmsource "Data
Source=.;Database=WebForms;Integrated Security=True" /wffmreadertype
"Sitecore.Forms.ConversionTool.Reader.SqlServerReader"
```

If the conversion tool is running normally from start to finish your output should look something like this:



Included in this information is the number of items processed and the time taken to complete the conversion process.

3. Execute the `DMS75_MigrateDefinitions.sql` script on your target Sitecore 7.5 primary reporting database. This script copies all the definition data from your source Sitecore 7.2 database to your target Sitecore 7.5 database. Definition data includes page event definitions, traffic types, visitor classifications and campaigns.

Note
Before you run the script update the `@sourceDatabase` reference at the top of the script, to specify the name of your source Sitecore 7.2 database.

For example:

```
DECLARE @sourceDatabase NVARCHAR(255)
SET @sourceDatabase = '[Sitecore_analytics7.2]'
```

4. The final step of the process is to synchronize the reporting database (SQL Server) with the collection database (MongoDB). For instructions on how to do this, see Chapter 5 *Rebuilding the Reporting Database*.

   Important
   It is strongly recommended that you do not start the synchronization of the reporting database until the conversion tool has completely finished the conversion process.

4.3 Messages and Notifications

When you run the conversion tool you might encounter the following error, warning, or notification messages.

Message	Type	Description
Cannot convert visit GUID. Failed to load location.	Error	It is impossible to convert visit data if the location information is corrupted. Therefore this visit will be dropped.
Failed to create visit GUID.	Error	This is a common error that informs you that the visit cannot be converted. The previous error includes a description about the problem that occurred during conversion.
`PageEvent` definition id: GUID does not exist.	Warning	This error means that the page event data is corrupted and that the definition item does not exist. Therefore a lot of page event information is lost such as `EventName`, `EventValue`, `IsGoal`.
Failed to obtain keyword by ID: GUID.	Warning	Relates to corrupted keyword data. If you get this error the keyword will be skipped.
Profile `Profile_Name` defined twice for visit GUID.	Warning	Profile with max total value will be converted. This error indicates inconsistencies in profile information.
Cannot convert `AutomationState` GUID `stateId` is empty.	Warning	Relates to a broken automation state record that does not correspond to an automation plan.
Key is present in cache of the sequence.	Warning	In general this means that the visitor is processed twice. This can happen if errors occurred during conversion.
Contact GUID classified as robot and will not be converted.	Warning	This warning occurs if exclude robots is switched on.
Cannot find workers with names:	Warning	The configured workers do not exist, check your configuration, and try again.

Chapter 5

Rebuilding the Reporting Database

The final step you need to perform in the database conversion process is to rebuild the reporting database. Follow these steps after the conversion tool has completely finished running and converting your historical data.

This chapter includes the following sections:

- Reasons for Rebuild
- Rebuilding the Reporting Database
- Post Rebuild Steps

5.1 Reasons for Rebuild

Rebuild of the reporting database is the re-processing of interactions that have already been aggregated into the reporting database for use by Sitecore reporting applications. You need to perform this as the final step of the conversion process to merge historical data with live data making this data available to Sitecore reporting applications.

Reasons for rebuilding the reporting database:

- After using the Sitecore Analytics Conversion Tool to populate the reporting database with analytics information from an earlier version of Sitecore.
- After having amended information in the collection database, in order to reflect the amendments in reports when looking at older data. Such amendment can be for example assigning channels to referring sites.
- If you have re-classified a search key word or traffic type, aggregated report data is not updated automatically.
- If the reporting database has been lost or is irrecoverably out of sync with the collection database, for example, due to a disaster or if the details of two contacts have been merged.
- In the *Executive Insight Dashboard* and other Sitecore reporting applications it is possible to reclassify data that has already been processed by the aggregation layer. This could cause the reporting database to become out of sync with the collection database.

Note
The process is semi-automated but also requires an administrator to attach, detach, or replace databases in SQL Server and also to modify some configuration files.

Note
To ensure that the reporting database contains the latest data changes, you need to rebuild it from time to time. Be aware that when you rebuild the reporting database, you overwrite its contents. For best results, we recommend that you use a clean copy of the *Sitecore_Analytics* database every time you perform a rebuild.

5.2 Rebuilding the Reporting Database

Follow the steps in this section to rebuild the reporting database.

5.2.1 Prerequisite Databases

In the `App_Config/ConnectionStrings.config` you should have the following databases already connected. These databases are required before you can start the rebuild process:

- Collection - MongoDB database. This contains converted historical data from your Sitecore DMS SQL Server database and recently collected live data from your website.

```
<add name="analytics" connectionString="mongodb://localhost/analytics"/>
```

- Primary reporting - SQL Server database. This contains all the live data collected on your website after upgrade to Sitecore 7.5.

```
<add name="reporting" connectionString="user
id=_sql_server_user_;password=_user_password_;Data
Source=_sqlserver_;Database=Sitecore_Reporting" />
```

- Secondary reporting - SQL Server database. This contains all the historical data and is used during the rebuilding the reporting database.

```
<add name="reporting.secondary" connectionString="user
id= sql server user ;password= user password ;Data
Source=_sqlserver_;Database= Sitecore_Reporting_Secondary" />
```

5.2.2 Rebuild Steps

To rebuild the reporting database:

1. Take a clean copy of the `Sitecore_Analytics` database from your Sitecore distribution to use as your reporting secondary database. Always use a clean copy for best results.
2. In SQL Server Management Studio, attach this database to your SQL Server instance. Use the name `reporting.secondary` in the connection string.

   Note
   The reporting and reporting secondary databases may occupy significant disk space and so may require significant forward planning of your storage requirements.

3. Add or edit the `reporting.secondary` connection string to point to the newly created database, for example:

```
<add name="reporting.secondary" connectionString="user
id= sql server user ;password= user password ;Data
Source=_sqlserver_;Database=Sitecore_Reporting_Secondary" />
```

   Note
   This step is mentioned earlier in Installing the Upgrade Package. This is to prepare for rebuilding the reporting database, so you do not need to perform this step twice.

4. If you are configuring a dedicated server, check that you have enabled the `history` processing pool in the relevant configuration file.

To enable history processing on a dedicated content delivery server, navigate to the following processing configuration file:

`App_Config/Include/Sitecore.Analytics.Processing.Aggregation.ProcessingPools.config`

Make sure the `<Enabled>` parameter is set to `true`:
`/configuration/sitecore/aggregationProcessing/processingPools/history/Enabled`

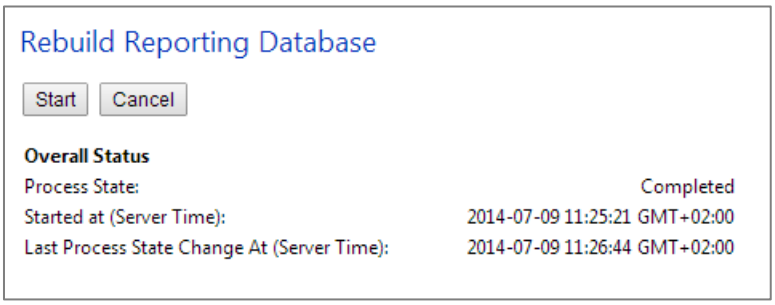
```
<history type="Sitecore.Analytics.Data.MongoDb.ProcessingPool.MongoDbProcessingPool,
  Sitecore.Analytics.MongoDb" singleInstance="true" >
  <param desc="connectionStringName">tracking.history</param>
  <Name>history</Name>
  <Enabled>true</Enabled>
</history>
```

5. Open the rebuild reporting database history processing page in a new browser window using the following path:

```
<sitename>/sitecore/admin/RebuildReportingDB.aspx
```

6. Click Start to begin rebuilding the reporting database (synchronization processing).

   The tool gives you feedback while it is processing until it has completed.



Note
When you run the rebuild process (synchronization) this erases any information already contained in the *Sitecore_Reporting_Secondary* database.

We recommend you use a clean copy of the database every time you rebuild the reporting database, as this speeds up the rebuild process, and uses fewer resources.

5.3 Post Rebuild Steps

After the rebuild process has finished:

1. Navigate to the `Website/App_Config` folder, and change the names in the `ConnectionStrings.config` file `Database=` attribute, so that the *reporting* connection string points to the *Sitecore_Reporting_Secondary* database which has now replaced the *Sitecore_Reporting* database.
2. Comment out the *reporting.secondary* connection string.

   Example connection strings after swapping the database names and commenting out the *reporting.secondary* connection string:

   ```
   <add name="reporting" connectionString="user
   id=_sql_server_user_;password=_user_password_;Data
   Source= sqlserver ;Database=Sitecore Reporting Secondary" />

   <!--
   <add name="reporting.secondary" connectionString="user
   id=_sql_server_user_;password=_user_password_;Data
   Source=_sqlserver_;Database=Sitecore_Reporting2" />
   -->
   ```

   This means that the xDB will now use the *Sitecore_Reporting_Secondary* database as its primary reporting database to collect new, live data from your website and that the *Sitecore_Reporting2* database is disabled until you need to enable it again the next time you rebuild the reporting database.

   Note
   Sitecore_Reporting2 is an example name you could use for the clean reporting database for the next rebuild the reporting database.

3. To verify that the rebuild has been successful check the information in the reporting database by opening the *Engagement Analytics* reporting application and view some reports. View *Latest Activity*, *Recent Visits*, the visits shown in this report should correspond to the historical data that came from the xDB MongoDB collection database.

   Note
   If you encounter problems, you can easily swap back connection strings to restore the previous version of the reporting database.