

Staging Module

Installation and Configuration Guide

Installation and configuration advice for administrators and developers



Contents

Chapter 1	Introduction	3
1.1	This Document	3
1.2	Features	3
1.2.1	Major differences between Staging for Sitecore 4 and Staging for Sitecore 6	3
1.2.2	Defining terms	3
1.2.3	Principle and relations	4
1.2.4	What triggers a Staging?	5
Chapter 2	Installation Guide – MS SQL Databases	6
2.1	Master Web Server	6
2.1.1	Set Up Publish Targets	6
2.2	Slave Web Server	8
2.2.1	Web.config changes	9
Chapter 3	Configuring the Staging Server	10
3.1	Add New Staging Servers	10
3.2	Creating Upload and Download Operations	13
Chapter 4	The Staging Console	15
4.1	Finding the Staging Console	15
4.2	The Overview	15
4.3	Table Operations Lists	16
Chapter 5	Frequently Asked Questions	17
5.1	Package Exception	17
Chapter 6	Administration Notes	18
6.1	Transfer New Files Only Functionality	18
6.1.1	Transfer New Files Only When Uploading	18
6.1.2	Transfer New Files Only When Downloading	18
6.1.3	Transfer New Files Only and Passive Mode	18
6.1.4	Delete Destination Files Before Upload Functionality	19
6.2	SOAP vs. FTP Uploads/Downloads	19
6.2.1	Advantages of the SOAP transfer mode	19
6.2.2	Advantages of the FTP transfer mode	19
6.2.3	Important Notes	19
6.2.4	Performance Comparison	20
6.3	Changing Staging Working Directory	20
6.4	Advanced configuring staging slave servers	21
Chapter 7	Troubleshooting	22
7.1	Is the StagingAgent scheduled task being run at all?	22
7.2	Check the Staging log file	22
7.3	Using SOAP: Is the web service properly installed on slave web servers?	22
7.4	Check Sitecore log file on Runtime	23
7.5	Check the language nodes	23
7.6	Logfile throwing "Error clearing cache: EOF in header"	23

Chapter 1

Introduction

The main purpose of the Sitecore Staging module is to update two or more Sitecore installations across a firewall.

1.1 This Document

This document contains the following chapters

- [Chapter 1 – Introduction](#)
- [Chapter 2 – Installation Guide – MS SQL Databases](#)
- [Chapter 3 – Configuring the Staging Server](#)
- [Chapter 4 – User Guide](#)
- [Chapter 5 – Frequently Asked Questions](#)
- [Chapter 6 – Administrator Notes](#)
- [Chapter 7 - Troubleshooting](#)

1.2 Features

The remainder of this chapter describes the features implemented by the Staging Module

1.2.1 Major differences between Staging for Sitecore 4 and Staging for Sitecore 6

Sitecore CMS 6 is by default capable of publishing to several databases. This is known as “publishing targets”, and a publishing target is simply the name and address of a Sitecore web database.

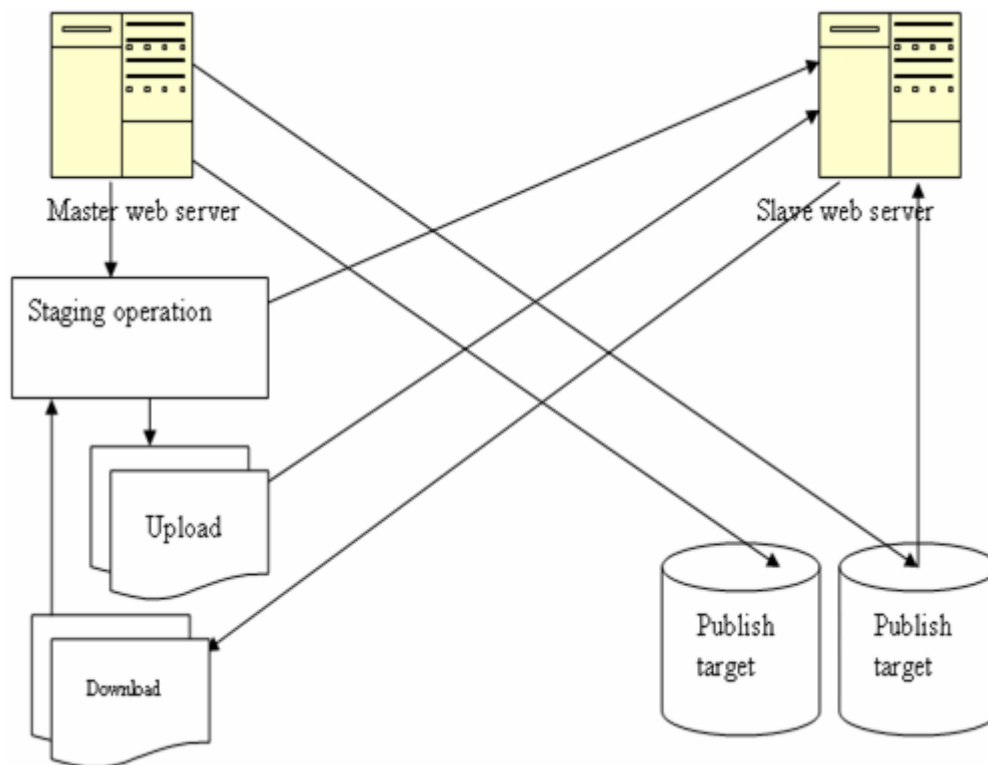
The Staging for Sitecore CMS 6 is therefore only a module that handles file transfers and emptying the cache of the slave Sitecore so that the website is capable of reflecting the web database update.

1.2.2 Defining terms

Term	Definition
Master Sitecore	The local Sitecore where all database updates are performed. The master Sitecore is usually installed inside the firewall.

Term	Definition
Slave Sitecore	A read-only Sitecore installation, usually installed outside the firewall, or in the DMZ. The slave Sitecore has no Sitecore client, and is only updated by the master Sitecore.
Staging operation	The process of updating a slave Sitecore database, uploading and downloading files to/from the slave Sitecore and refreshing the slave Sitecore web server cache.
File operation	The process of uploading or downloading files to/from the slave Sitecore.
Slave web server	The IIS where the slave Sitecore is installed. A Staging operation always begins with clearing the cache of the slave web server.
Publishing targets	Used by the Staging as web database targets. A publishing target is the name of an SQL server type which Sitecore publishing writes its web data.

1.2.3 Principle and relations



The Staging module is installed on the master Sitecore.

The number of publishing targets is defined on the master Sitecore. One or more of these publishing targets points to the web database of one or more Slave web servers.

One or more staging operations are defined on the master Sitecore.

The operations define the names of one or more slave web servers.

For each staging operation, the number of file operations is defined, each upload or download filed to or from the slave web server is defined in the Staging operation.

Note: The writing of data to the publishing target has nothing to do with the Staging operation as such. A publishing target is the default way of writing web data to a database in Sitecore CMS 6. The difference between a standard Sitecore CMS 6 installation and a staged environment is therefore only in the fact that you define not a single, but several web databases. See also the following chapter [“Major differences between Staging for Sitecore 4 and Staging for Sitecore 6”](#).

1.2.4 What triggers a Staging?

The Sitecore Publish will publish data to all selected languages and publishing targets in turn. If you have 2 languages and 2 publishing targets, 4 publish operations will be executed.

For each publish operation, an event is fired. Staging hooks into this event, and examines whether a Staging operation should be executed or not (this is defined by the Staging Server).

If a staging operation is found, a series of work files are written to the master Sitecore. The work files contain commands to clear web cache and files transmission.

These work files are picked up and executed by the scheduled task. If an execution of a work file is successful, it is deleted; otherwise not. This ensures that a failed execution will be retried until successful.

Note: if you want Staging operations to be executed for every language or/and publish target, set the Target language or/and Publishing target fields of the Staging Server to empty value.

Chapter 2

Installation Guide – MS SQL Databases

This chapter deals with the process of installing the Staging module using MS SQL Databases.

2.1 Master Web Server

Install the **staging_master** package on your master web server. The Staging module is distributed as a usual Sitecore package. You can install it using the Installation Wizard available via the Sitecore desktop link *Sitecore » Development Tools » Installation Wizard*.

The package installs the `/App_config/Include/staging.config` file which contains the module configuration settings.

2.1.1 Set Up Publish Targets

Note: it is highly recommended to use `ServerConnections.config` when configuring Staging as it is not always possible to reach a remote database via the `FileConnections.config`.

In `/App_Config` directory edit the `ConnectionStrings.config` file. Set up the following:

Add a connection to a remote web database (the target database for staging) to the `/connectionStrings` section. In our example we will just copy the string for the web database and edit it:

```
<add name="webtarget" connectionString="user id=user;password=password;Data Source=server;Database=database" />
```

Select the connection definition name with care. For example, if you have one slave web server per publishing target, use the slave server name as a key.

If you have several slave web servers per publishing target, use the SQL server name as a key.

In `web.config`, set up the following:

1. Add a database node to the `/Sitecore/databases` section. This can be done by copying the `/Sitecore/databases/database[id="web"]` node and pasting it below. Change the database id to the connection id defined in the previous step:

```
<database id="webtarget" singleInstance="true"
type="Sitecore.Data.Database, Sitecore.Kernel">
  <param desc="name">$(id)</param>
  <securityEnabled>true</securityEnabled>
  <dataProviders hint="list:AddDataProvider">
    <dataProvider ref="dataProviders/main" param1="$(id)">
```

```

    <disableGroup>publishing</disableGroup>
    <prefetch hint="raw:AddPrefetch">
      <sc.include file="/App_Config/Prefetch/Common.config">
        </sc.include>
      <sc.include file="/App_Config/Prefetch/Web.config">
        </sc.include>
      </prefetch>
    </dataProvider>
  </dataProviders>
  <proxiesEnabled>false</proxiesEnabled>
  <proxyDataProvider ref="proxyDataProviders/main" param1="$(id)">
  </proxyDataProvider>
  <cacheSizes hint="setting">
    <children>100KB</children>
    <credentials>100KB</credentials>
    <data>20MB</data>
    <items>10MB</items>
    <parents>100KB</parents>
    <paths>500KB</paths>
  </cacheSizes>
</database>

```

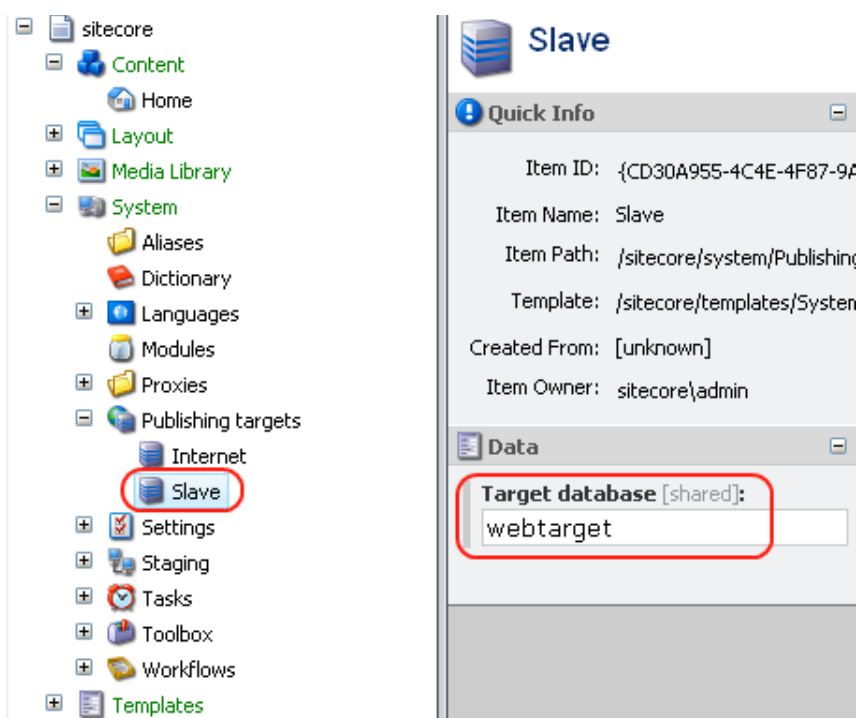
2. [Optional] To improve cache clearing performance on slave server, you should add the history engine to the database definition and the following hook to web.config file:

```

<sitecore><configuration>...
<database id="webtarget">
...
  <Engines.HistoryEngine.Storage>
    <obj type="Sitecore.Data.$(database).$(database)HistoryStorage,
Sitecore.Kernel">
      <param connectionStringName="$(id)" />
      <EntryLifeTime>30.00:00:00</EntryLifeTime>
    </obj>
  </Engines.HistoryEngine.Storage>
</database>
...
<hooks>
  <hook type="Sitecore.Modules.Staging.InitializeEngines, Staging.Kernel" />
</hooks>
...
</configuration></sitecore>

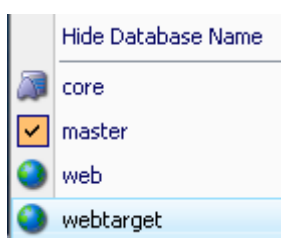
```

3. Finally, open the Sitecore shell and add a node to the **/system/publish targets** section:



Enter the connection id defined above in the *Target database* field.

You should see the newly added database in the database list:



Repeat these steps for each of your publishing targets.

2.2 Slave Web Server

Perform the following actions on the slave web server:

1. Install the Staging web service (**staging_slave** package) on each slave web server. Refer to the following link to [read about package installation in Sitecore CMS](#).
2. [Optional] To improve cache clearing performance on slave server, you should add history engine to the web.config file for the production database (see 2.2.1 chapter in this guide).
3. If you intend to use FTP as the file transferring method, you should install an FTP server on the slave web server.
4. Disable the UploadWatcher in the web.config of the Slave Server:

```
<!-- add type="Sitecore.Resources.Media.UploadWatcher, Sitecore.Kernel"
name="SitecoreUploadWatcher"/ -->
```


This change is conditioned by the following fact: if the Slave server has the Master database and users put or upload files to the **/upload** folder, the UploadWatcher starts to create media items in the Master database which is unnecessary.

Proceed to the Configuring Staging Server section when done.

2.2.1 Web.config changes

The staging module has the ability to clear not the entire cache, but only the fragment of the cache for the data that were changed after publishing from the master server. In order to enable this feature, the following changes need to be made to the web.config file of the slave server.

1. Open web.config file;
2. Go to configuration>sitecore>databases section;
3. Select the production database element e.g. <database id="web" singleInstance="true" type="Sitecore.Data.Database, Sitecore.Kernel">;
4. Add the following code under the database element:

```
<Engines.HistoryEngine.Storage>  
  <obj type="Sitecore.Data.%(database).%(database)HistoryStorage, Sitecore.Kernel">  
    <param connectionStringName="%(id)" />  
    <EntryLifeTime>30.00:00:00</EntryLifeTime>  
  </obj>  
</Engines.HistoryEngine.Storage>
```

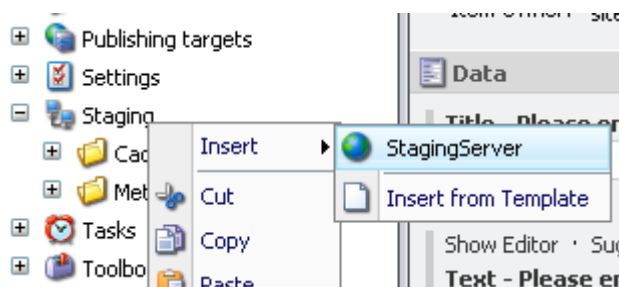
Chapter 3

Configuring the Staging Server

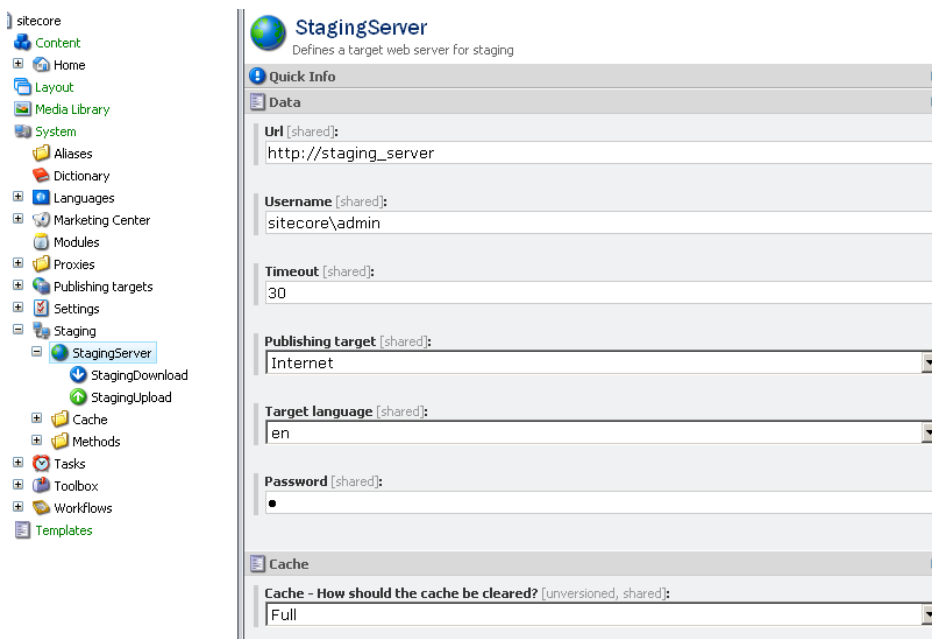
When all web.config changes are done, proceed to configuring the Staging module in the Sitecore CMS interface.

3.1 Add New Staging Servers

Navigate to the `/sitecore/system/staging` item, right-click on it and select **New » Staging Server**:



Each Staging Server item represents one slave server (staging target). The list of the fields provided by this item is shown below.



The Data Section

Field	Description	Sample value
Url	The URL of the slave web server SQLite note: the address must have the trailing slash.	http://staging_server/
Username	The name of any user on the slave web server. Be sure to use full user name (domainname\username).	sitecore\admin
Password	The password for the Sitecore CMS client of the specified user	56v23
Timeout	The timeout in seconds for the Staging web service	15
Publishing target	The publishing target that, combined with the target language, triggers a staging operation. See the What triggers a Staging? Chapter for details. MS SQL note: the publishing target specified here should reflect the actual target database.	webtarget
Target language	The language that, combined with the publishing target, triggers a staging operation. See the What triggers a Staging? Chapter for details. Note: this field is empty by default.	en

The Cache Section

Field	Description
Cache	Determines how the cache should be cleared: <ul style="list-style-type: none"> Partial: Fast. Clears only changed data in caches. Will work only when the history engine is activated for production database on both master and slave servers. Please check optional configuration changes for master and slave servers. Full: Slow. Clears all caches.

The FTP Section

Use this section to configure the FTP connection if you plan to transfer files via FTP.

Field	Description	Sample value
Server	The name of the FTP server. Enter only the name itself in this field without any slashes or an ftp prefix.	ftpsrvr

Field	Description	Sample value
Port	The FTP server port. Default is 21	880
Username	Username for the FTP server.	ftpuser
Password	Password for the specified user.	ftppass
Passive transfers	If checked, the FTP server will use passive mode to transfer data	unchecked
Restart on errors	If checked, the FTP server will restart the FTP connection automatically on connection errors	checked
Delete destination file before upload	Deletes the existing destination files before uploading new ones. This flag also eliminates the “Passive transfers” feature. For more information, see the Delete destination file before upload functionality chapter.	checked
Timeout	FTP inactivity time in seconds before the connection is aborted.	10
Encoding (FtpEncoding)	Encoding to use when communicating with the FTP server. This affects the handling of files and directories with national (non-ASCII) characters in names. Note: When the field is empty, ASCII encoding is used for communication.	

The Finalize Section

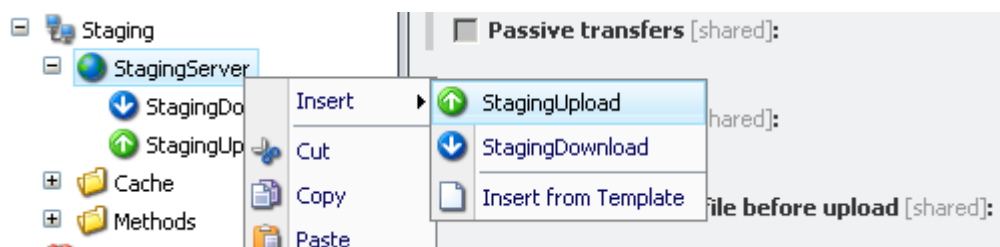
This section allows to specify some links, accessing which will report a successful staging operation or an error.

Field	Description
Call URL	If checked, the staging will call one of the 2 URLs described below.
Url on Success	The URL to call if all of the items scheduled for staging were transferred successfully.
Url on Error	The URL to call if one of the items scheduled for staging was not transferred successfully.

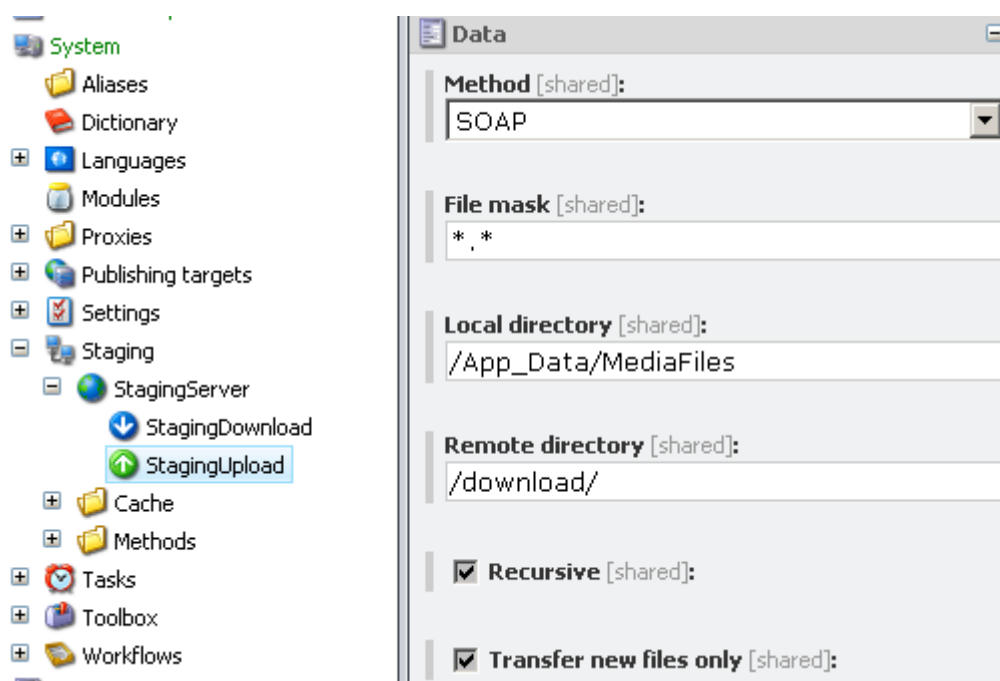
3.2 Creating Upload and Download Operations

In order to upload files to the slave server or download files from it you should define upload and download operations.

To create upload or download operation, right-click on the Staging Server item and select **New » StagingUpload/StagingDownload**.



Let us take a look at the StagingUpload item:



Below is the description of the item's fields (fields are the same for StagingUpload and StagingDownload items).

Field	Description
Method	<p>The preferred file transfer method:</p> <p>FTP: Use the FTP server defined in the StagingServer item.</p> <p>SOAP: Use the Staging web service.</p> <p>See also chapter SOAP vs. FTP Uploads/Downloads.</p> <p>Default value: SOAP.</p>



Field	Description
File mask	<p>The pattern of files to get from the source folder. Usual shell patterns are used: question mark (?) means any single symbol, asterisk (*) means any number of symbols. A mask containing an asterisk only (*) means every file. Another example is “*.jpeg“. This will get all the images with the JPEG extension in the directory.</p> <p>The file mask does not affect any directories. Even if you choose a file mask other than '*', all directories will be uploaded if you choose “Recursive”.</p>
Local directory	<p>Directory on the master web server. The path can be physical or logical.</p> <p>Examples:</p> <p>/App_Data/MediaFiles/upload/ D:\Sitecore\upload\</p>
Remote directory	<p>Directory of the slave web server. The directory can only be logical.</p> <p>Example: /upload/</p>
Recursive	<p>If true, Staging will also search for files in subfolders</p>
Transfer new files only	<p>Also known as UseLastUpdate.</p> <p>If true, only files which were added since the last publish will be transferred.</p> <p>See also Transfer new files only functionality.</p>

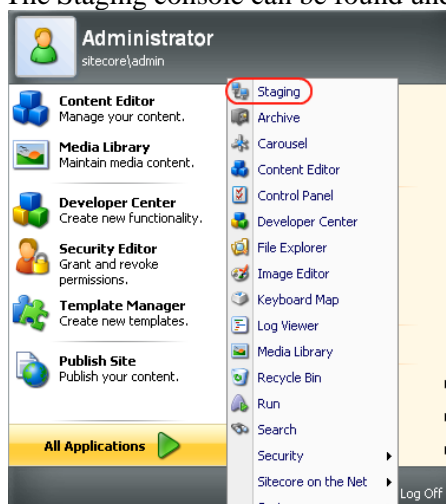
Chapter 4

The Staging Console

This chapter details the screens found in the staging console

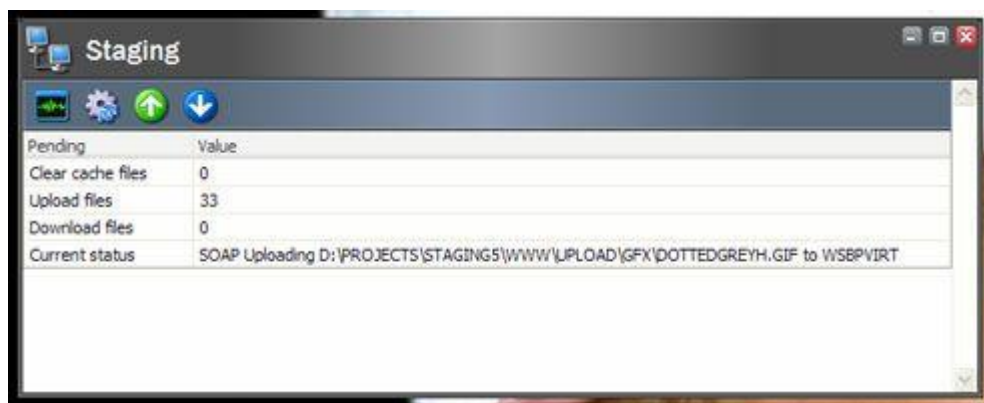
4.1 Finding the Staging Console

The Staging console can be found under **All Applications » Staging**:



4.2 The Overview





The Staging console shows an overview of the current activity and pending staging operations:



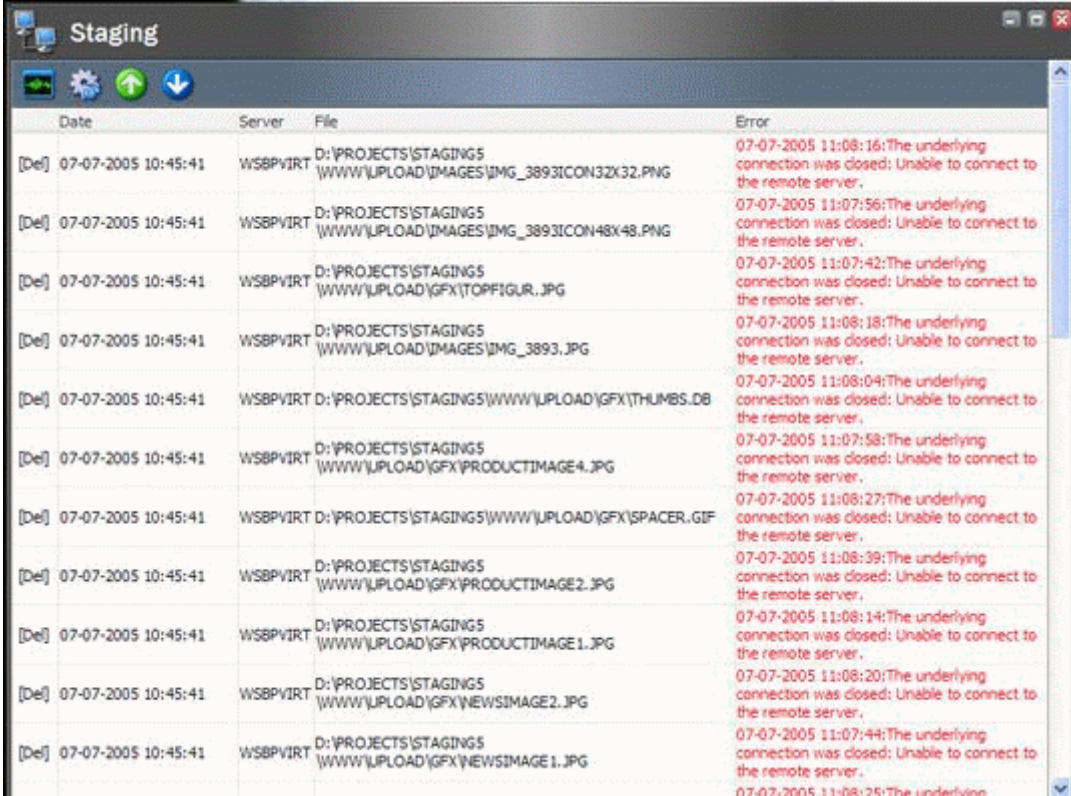
Pending	Value
Clear cache files	Number of clear cache operations pending
Upload files	Number of files to be uploaded
Download files	Number of files to be downloaded
Current status	The current activity. Will be "idle" if no activity is currently present

4.3 Table Operations Lists

Furthermore, you can get a list of all operations by clicking on the buttons on the tool bar:

Button	Action
	View status window (shown above)
	View pending clear cache files
	View pending upload files
	View pending download files

Here is an example of a window showing pending upload files:



Date	Server	File	Error
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\IMAGES\IMG_3893\CON32X32.PNG	07-07-2005 11:08:16:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\IMAGES\IMG_3893\CON48X48.PNG	07-07-2005 11:07:56:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\TOPFIGUR.JPG	07-07-2005 11:07:42:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\IMAGES\IMG_3893.JPG	07-07-2005 11:08:18:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\THUMBS.DB	07-07-2005 11:08:04:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\PRODUCTIMAGE4.JPG	07-07-2005 11:07:58:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\SPACER.GIF	07-07-2005 11:08:27:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\PRODUCTIMAGE2.JPG	07-07-2005 11:08:39:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\PRODUCTIMAGE1.JPG	07-07-2005 11:08:14:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\NEWSIMAGE2.JPG	07-07-2005 11:08:20:The underlying connection was closed: Unable to connect to the remote server.
[Del] 07-07-2005 10:45:41	WSBPVIRT	D:\PROJECTS\STAGINGS5\WWW\UPLOAD\GFX\NEWSIMAGE1.JPG	07-07-2005 11:07:44:The underlying connection was closed: Unable to connect to the remote server.
			07-07-2005 11:08:25:The underlying

The text in red indicates that there has been an error in the upload. To delete the work file (not the actual file to be uploaded) click on the [Del] button.

Chapter 5

Frequently Asked Questions

This chapter contains answers to frequently asked technical questions concerning the Staging module.

5.1 Package Exception

Q:

Why is the packager throwing 'UnauthorizedAccessException: Access to the path <*.dll> is denied'?

A:

Delete any files from /bin. Install and ensure ASP.NET (ASPNET or NETWORK SERVICE) has full control of that directory, which is used by the packager to avoid recycling the ASP.NET worker process too early during package installation process.

Chapter 6

Administration Notes

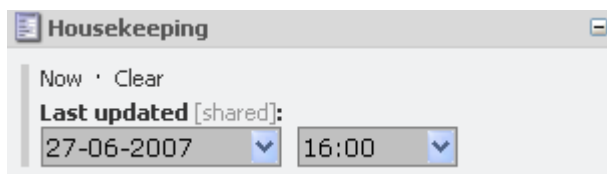
This chapter outlines notes useful to site administrators

6.1 Transfer New Files Only Functionality

This feature allows you to reduce the number of uploaded or downloaded files.

6.1.1 Transfer New Files Only When Uploading

The function will store the time of the last transfer in the Last updated section of the StagingServer:



When the next uploading is carried out only files that are newer than this date will be uploaded.

If you wish to upload all files again, simply delete the value of Last updated field and save the item. Next time you use the module all files will be uploaded again.

6.1.2 Transfer New Files Only When Downloading

Since the time on the FTP can be different from that of your Sitecore CMS server, storing the time of the last download operation will have no effect.

Instead, when downloading, each downloaded file will have its timestamp changed to that of the file on the FTP server.

Next time the module attempts to download the same file, the dates are compared and the file is downloaded only if the file on the FTP server is newer than the file on the master server.

6.1.3 Transfer New Files Only and Passive Mode

There is a minor difference between Transfer New Files Only functionality and Passive Mode transferring.

Passive mode is controlled by the FTP server and will reject uploading or downloading of files unless their size has been changed. The content and modification dates of the files are ignored.

Transfer New Files Only functionality compares the file modification dates, not file sizes.

Take a look at the following example:

Consider a .gif image representing a blue bullet. A user changes the color from blue to red. The size of the image will not be changed, but the modified date will.

In passive mode, the file will not be uploaded because the file size is the same.

6.1.4 Delete Destination Files Before Upload Functionality

This feature will delete any existing files on the FTP server before uploading new files.

This eliminates the replace/append problem on certain FTP servers. The problem has the following symptoms:

If the new file is smaller than the original file it is written on disk starting from the beginning of the original file, leaving the last part of the original file in the end of the new one.

Also, some FTP servers will only append new text to the end of the original file, leaving the first part intact. This is more frequent when using passive transfer mode.

Please note that if you set this option to true, the Staging module will run substantially slower.

Only use this setting if you experience any of the problems described above.

Also, when using this feature, “passive mode” has no effect on the existing files.

6.2 SOAP vs. FTP Uploads/Downloads

The Staging module has a feature for transmitting files via a web service rather than FTP server. The pros and cons of the two modes are described below.

6.2.1 Advantages of the SOAP transfer mode

- No need for setting up an FTP server on the slave web server
- No need for opening additional ports. The web service uses the HTTP or HTTPS port.
- Secure transmission is possible by using an HTTPS connection.
- Faster transmission of files smaller than 1.5 Mb.

6.2.2 Advantages of the FTP transfer mode

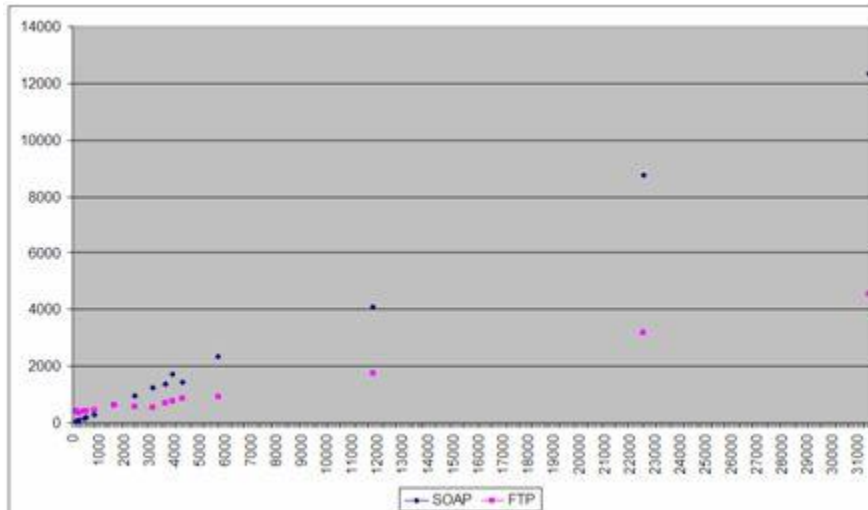
- Standard way of transmitting files.
- Faster transmission of files larger than 1.5 Mb.

6.2.3 Important Notes

If you need to update files in the /bin folder from Master to Slave server, you should do this either manually via FTP or use FTP upload mode but not SOAP mode. If the SOAP mode is used, Sitecore uploads files using a remote web service that is located on the Slave server. The architecture of ASP.NET and .NET Framework is such that if any DLL is updated in the /bin folder of the Slave server, ASP.NET worker process is restarted which makes the Slave server inaccessible for some time and breaks the upload process. When the FTP mode is used instead of SOAP mode, all files are uploaded without problems but the ASP.NET worker process of the Slave server is restarted anyway. If there are Extranet logged in users, they will lose their session because of the ASP.NET restarting.

6.2.4 Performance Comparison

The graph below shows the file transmission time against file size. X-axis represents file size in bytes; Y-axis represents transmission time in milliseconds:



Uploading of files smaller than 1.5 Mb is faster with the SOAP mode, but the FTP mode becomes increasingly faster for files larger than 1.5 Mb.

6.3 Changing Staging Working Directory

After the Staging installation a folder called **/Sitecore modules/staging/workdir** is created and used as the module's working directory. The directory contains the following:

- Status.xml – a file containing the current status of the module.
- stagingYYYYMMDD.log.txt - staging log files.
- One directory for each StagingServer configuration.

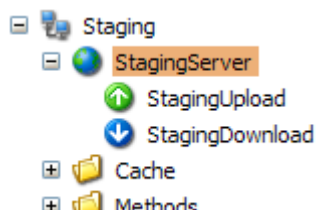
Each staging server configuration directory contains the following:

- Settings.xml – a file containing all StagingServer settings.
- Cache – a folder containing work files for clearing cache.
- Download - a folder containing work files for downloading operations.
- Upload - a folder containing work files for uploading operations

Below is the example of an upload work file:

```
<?xml version="1.0" encoding="utf-8"?>
<root server="StagingServer" doctype="upload" action="put"
initiator="StagingServer.StagingUpload" method="FTP">
  <local>D:\SITECORE\WEBSITE\APP_DATA\MEDIAFILES\FILES\STAGING5_4.JPG</local>
  <remote>/upload/STAGING5_4.JPG</remote>
</root>
```

Work files are created in the appropriate folders for every upload, download and cache clearing operation defined under the Staging Server item:



The Staging agent defined in the <scheduling> section of web.config monitors those folders and executes the tasks defined in the work files. If a task completed successfully, the corresponding work file is deleted. Otherwise the file is left unchanged and is processed upon the next agent iteration.

To change the working directory, add the following to your /Sitecore/settings section of your web.config:

```
<Setting.name="stagingWorkDir" value="_yourpath_" />
```

Change “yourpath” to an appropriate logical or physical path. Make sure that the ASPNET user (Network Service user in Windows 2003) has 'write' permission on the /**sitecore modules** folder.

6.4 Advanced configuring staging slave servers

The slave servers often may be configured with non-default values e.g. web database has non standard name or content delivery server doesn't contain any shell part of the Sitecore. In this case the default context references of the Staging module should be changed. For this purpose the following settings can be added to the web.config file of the slave servers:

- stagingWebDatabase - specify the name of the web database of the slave server if it differs from the default one (web);
- stagingSiteContext – specify the name of the site to be used as context site; this is needed when there is no shell part on the content delivery server (there is no shell site).

Chapter 7

Troubleshooting

In this chapter is a list of troubleshooting tips in case the items and files are not synchronized when publishing.

7.1 Is the StagingAgent scheduled task being run at all?

Open the sitecore log file and check that it contains entries for the **StagingAgent**. If the logfile doesn't contain any StagingAgent entries (or only very few of them), check the `<frequency>` setting in the `<scheduling>` section of the master web server's web.config file (as mentioned in the Staging module installation guide). The default frequency is `<frequency>00:10:00</frequency>` meaning that it can last up to 10 minutes after invoking the publish action until the StagingAgent task (which performs the file upload) is invoked by Sitecore.

7.2 Check the Staging log file

Open the log file in `"/sitecore modules/staging/workdir"` on the master web server (or the directory specified in the "StagingWorkDir" setting in web.config). If no logfile is generated, either the module is not installed correctly or the asp.net process doesn't have write access to the working directory.

The logfile contains useful information about file synchronisation including error messages that can indicate various problems (such as being unable to contact the slave server due to the slave server's host name not being found, or being unable to call the web service on the slave web server).

7.3 Using SOAP: Is the web service properly installed on slave web servers?

Try to access the following URL on each of the slave web servers (replace "example.com" in the address with the hostname/URL that you have specified for the slave in the StagingServer item in Sitecore):

<http://example.com/sitecore%20modules/staging/service/api.asmx>

This should bring up a list of available SOAP methods.

If this is not the case, check that the file "api.asmx" exists in the `"/sitecore modules\staging\service"` folder on the slave server. If the file does not exist, make sure the ASPNET (and possibly IIS_XXX) user (**Network Service** user in Windows 2003) has write access to the `"/sitecore modules"` folder and try to reinstall the Staging_slave.zip package.

7.4 Check Sitecore log file on Runtime

Check the Sitecore logfile on the runtime server to see if any errors are logged regarding staging.

No images/files being uploaded?

If the log file doesn't contain entries about any file/image uploads, try the following:

- Check the scheduling frequency (as mentioned above)
- If changing the scheduling frequency doesn't work, go to the "Housekeeping" section of the "StagingServer" item that is parent to the "StagingUpload" that isn't working as expected. Remove the value in the "Last updated" field, save the item and republish.
- Finally go to the "StagingUpload" item and temporarily remove the checkmark from "Transfer new files only" and republish.

7.5 Check the language nodes

This information applies to Sitecore CMS 6:

- Check that the "Iso Code" field is filled out correctly for all the languages below (all the nodes below /sitecore/system/languages in the master database).

The Staging module use the information in the "Iso Code" field when determining if a publish operation should trigger a staging action. If the "Iso Code" is not filled out (or is filled out incorrectly), the staging will never be triggered.

7.6 Logfile throwing "Error clearing cache: EOF in header"

You probably have the dataMigrationWizard installed. If you do, the staging will not work until you disable it.

You need to disable the following section in your web.Config the ClearCache call probably will work again.

```
<webServices>
  <soapExtensionTypes>
    <add type = "
TransferProtocol.SoopExtensions.SoopArchiveExtension,TransferProtocol " priority
= " 1 " group = " 0 " />
  </soapExtensionTypes>
</webServices>
```