# Geocortex

## Site Migration Application

### Summary

The site migration application will move a Geocortex Essentials REST Elements site from one environment to another environment, replacing environment-specific strings such as URLs in the process. The migration may occur between environments on the same computer or between environments on different computers.

It is anticipated that sites will be migrated to and from environments that are relatively similar. In fact, the primary use case for this migration application is to facilitate migration of sites from nearly identical environments on one server to another server. That said, the script is flexible enough to be used for migration of sites between Essentials instances on the same server, or even from a named instance on one server to an alternately named instance on a target server.

### Architecture

#### Console Application

The site migration application has been built as a Microsoft .NET 4 console application wrapper around a DLL containing the application logic. The application is named SiteMigrationApp.exe. The DLL is named Geocortex.Services.SiteMigration.dll. The separation of console application from DLL was done to enable future user interface wrappers to be written around the site migration functionality.

The console application expects to be run with three parameters, in this order:

* ID of the site to migrate
* ID of the environment to migrate from
* ID of the environment to migrate to

See the configuration section below for details on these parameters.

#### File List

The full list of files necessary for the operation of the site migration script is as follows:

* NLog.config
  + Logging configuration file which defines the format of the SiteMigrationApp logs, the logging detail level, and the location of the output log files. This is a standard NLog configuration file, and can be modified as necessary to suit logging requirements. See <http://nlog-project.org/> for details.
* NLog.dll
  + Logging library reference.
* NLog.xml
  + Logging library documentation. Not required for operation of the application but useful for development purposes.
* Geocortex.Services.SiteMigration.dll
  + Compiled application logic.
* SiteMigration.exe
  + Compiled wrapper application around the DLL logic.
* SiteMigration.exe.config
  + Application configuration file, necessary to reference the location of the site migration configuration file.
* SiteMigrationConfiguration.xml
  + Site migration configuration file. The location and name of this file is controlled by the SiteMigration.exe.config file, but by default is set to SiteMigrationConfiguration.xml in the current directory (current directory set by the user’s current directory when executing the SiteMigrationApp.exe file).

#### Source Code

In addition to the compiled application, the source code is included as a Visual Studio 2010 solution. Refer to the source code and comments therein for details.

### Flow of Events

Upon invocation of the site migration application, the site migration configuration is read in from the xml file as specified in the SiteMigration.exe.config. The parameters supplied to the SiteMigrationApp.exe application are matched up by ID with Site and Environment entries in the configuration file. If no matching site configuration can be found matching the supplied Site ID, a new site configuration is created which will copy just the site folder with no dependent files. If no environment configuration can be found matching either the source or destination environment IDs supplied, the application will terminate and an error will be logged to the console.

Prior to site migration, the site in the target environment is backed up. The backup location is configured in the environment configuration section of the xml file. The site directory plus all configured site dependencies are backed up into a directory named with a timestamp plus the ID of the site being backed up. An example backup directory name would be “2012-03-15 11 35 30 1234 – MySite”. If the site does not yet exist in the target environment, the backup directory will still exist but will not contain any files. Dependent resources may still exist in the target environment even if the site does not yet exist (such as shared workflows or print templates) – if this is the case then the dependencies will be backed up even if the site folder itself is not.

The site folder is then copied from the source environment to the target environment. The location of the site folder is constructed from the Essentials base path configured in the xml file for the environment, plus the instance name, plus the hard-coded path “REST Elements\Sites”, plus the site ID.

If the source site folder does not exist or cannot be accessed, the SiteMigrationApp will terminate.

The site resources as configured in the xml file are then copied in order from the source environment to the destination environment.

For files both in the site folder as well as the resource dependencies, if the file extension matches a configured “token” file extension pattern, the environment tokens will be replaced during the copy phase. This is to allow for environment-specific paths in configuration xml files, workflows, reports (and other text file types) to be automatically modified during migration.

If at any point an error occurs copying either the site directory or any dependent resources, the application will automatically restore the files which had been backed up.

Note that the site backups are not automatically deleted at any point by the SiteMigrationApp. Management of the contents of the site backup directories is the responsibility of the client.

Also note that when restoring a site from a backup, any new files which had been added to the target environment are not removed. If necessary to remove any new files, this must be done manually. For example:

* A site migration contains a new workflow in the source environment
* The site migration script is run and this new workflow is copied to the destination environment
* A site resource fails to copy, triggering automatic restoration of the original site in the destination environment from backup

In this case, the new workflow still exists in the destination environment

### Configuration

#### SiteMigrationApp.exe.config

The SiteMigration.exe.config contains configuration only for the location of the main Site Migration Configuration XML file. Here is the default configuration:

<appSettings>

<add key="ConfigurationPath" value="SiteMigrationConfiguration.xml"></add>

</appSettings>

The default setting references the SiteMigrationConfiguration.xml file in the user’s current directory. This may be changed as necessary to reflect an alternate relative-path configuration file, or it can reference a fully-qualified name of a migration configuration file instead.

#### NLog.config

The NLog.config file controls the location and verbosity of the logs generated by the site migration application. The default setting writes extremely verbose logs to a logfile named for the current date, and writes less verbose log messages to the console. This can be changed as necessary – refer to <http://nlog-project.org/> for details on modification of this file.

Unless there are disk space considerations, the log level of the file logger should be kept at “Trace” to enable debugging of any migration issues that are encountered. The size of the trace log will vary by the number of files copied, but should be around 100kB per migration as a general rule.

#### SiteMigrationConfiguration.xml

This is the main configuration file for the site migration. It is broken down into three distinct sections – TokenFileExtensions, EssentialsEnvironments and Sites.

##### TokenFileExtensions

The TokenFileExtensions section allows you to define which file types will be searched for environment-specific strings such as ArcGIS Server URLs or Essentials URLs. The default setting for this section is as follows:

<TokenFileExtensions>

<TokenFileExtension>.xml</TokenFileExtension>

<TokenFileExtension>.xaml</TokenFileExtension>

<TokenFileExtension>.rpx</TokenFileExtension>

</TokenFileExtensions>

Each TokenFileExtension defines a file type in which tokens will be replaced. All files copied as part of the site migration process which have a file extension matching one of the TokenFileExtensions will go through an extra filter where environment-specific strings are replaced. The environment-specific strings which are replaced are defined in the individual environment configuration sections, described below.

Additional file extensions can be added, but ensure that they are only text file formats (not binary file formats such as images). If a binary file format is included, the behavior may be undesirable and may result in the final binary file in the destination environment becoming corrupted.

Each file extension must begin with a period – for example, “.txt” rather than just “txt”.

##### EssentialsEnvironments

The EssentialsEnvironments section allows for definition of many separate Essentials instances, which may reside on different servers. An example configuration of a single EssentialsEnvironment is as follows:

<EssentialsEnvironments>

<EssentialsEnvironment>

<ID>trainingVM</ID>

<DisplayName>Essentials 3.7.0 on Latitude-traini</DisplayName>

<BasePath>\\latitude-traini\c$\Program Files (x86)\Latitude Geographics\Geocortex Essentials</BasePath>

<DriveRoot>\\latitude-traini</DriveRoot>

<BackupBasePath>\\latitude-traini\c$\Program Files (x86)\Latitude Geographics\Geocortex Essentials\370\REST Elements\SiteBackups</BackupBasePath>

<Instance>370</Instance>

<EnvironmentTokens>

<EnvironmentToken>

<ID>EssentialsInstancePath</ID>

<Value>Geocortex/Essentials/370/REST</Value>

</EnvironmentToken>

<EnvironmentToken>

<ID>ArcGISServer</ID>

<Value>http://latitude-traini/ArcGIS/rest/services</Value>

</EnvironmentToken>

</EnvironmentTokens>

</EssentialsEnvironment>

</EssentialsEnvironments>

A few of the settings for an EssentialsEnvironment represent paths to directories. It is important to ensure that the user account running the site migration script has full access to all directories referenced in the EssentialsEnvironments settings, or the site migration may fail.

The individual settings for an EssentialsEnvironment are:

* ID: A unique identifier for this instance. There should not be more than one EssentialsEnvironment with the same name, or the site migration script may not recognize the correct environment when performing a migration. It is suggested that this ID not contain any spaces or special characters, since it must be supplied on the command-line to the SiteMigrationApp.exe program.
* DisplayName: A friendlier name for this EssentialsEnvironment. This setting has two purposes. First, it is used by the application to reference the EssentialsEnvironment in the output logs, where appropriate. Second, it is a convenient way for the site migration administrator to identify environments which may have been assigned more cryptic IDs.
* BasePath: The base installation directory of Geocortex Essentials in this environment, excluding the instance name. It is suggested that UNC pathnames be used to allow the site migration script to work across different computers. The base path is used to locate the primary directory of the site being migrated, by appending to the base path the value of the Instance property (described below), plus “REST Elements\Sites”, plus the ID of the site.
* DriveRoot: The root or UNC path to the drive on which the Essentials environment resides. This DriveRoot is used to resolve site dependencies which are absolute rather than relative to the site. For example, if a site has a dependency on a file located outside of the Essentials installation directory, for example in the Silverlight Viewer’s resources directory, it may be better to configure that dependency as an absolute path relative to the DriveRoot rather than as a path relative to the Site directory itself. See the Site ResourceDependency configuration description below for more details on how this DriveRoot is used.
* BackupBasePath: The path to the directory where backups for Sites from this Essentials environment will be saved. This may be a local path (e.g. beginning C:\) or a UNC path as required.
* Instance: The instance name of Geocortex Essentials. For a basic (single instance) installation of Essentials, this instance will be **Default**. In this case, the Instance may be omitted from the configuration – if omitted then the value of **Default** will be used automatically.
* EnvironmentTokens: Defines the list of strings which should be replaced when migrating sites to this environment. Each EnvironmentToken has both an ID and a Value – the ID allows strings to be matched up across environments, and the Value defines what should actually be replaced. For example, a dev and a test environment may define separate ArcGIS Server URLs – dev.maps.domain.com/arcgis/rest and test.maps.domain.com/arcgis/rest. In configurations for both the dev Essentials environment and the test Essentials environment there should be an EnvironmentToken with an ID of “ArcGISServerURL” (or similar). It is not important what ID is chosen, so long as the IDs match up across environments. During site migration from dev to test, the script will find all of the instances where dev.maps.domain.com/arcgis/rest is mentioned, and will replace it with test.maps.domain.com/arcgis/rest.

It is important to be aware of the implications of how the EnvironmentTokens are configured and ordered. For each file whose extension matches one of the TokenFileExtensions (described above), a search and replace will be done for each matching EnvironmentToken in the order that the tokens appear in the XML configuration document (for the source environment, if the order of the tokens differs). If certain environment tokens that you expect to be replaced during migration are not actually being replaced, consult the log file which outputs specific details about the token replacement process as it occurs.

If there is an EnvironmentToken in one environment but not in another (matched up by ID), when a site is migrated between those environments those unmatched tokens will be ignored. Any ignored (unmatched) tokens will be output to the log during migration, so check there if any inconsistencies are observed.

##### Sites

The Sites section defines external resources that a site is dependent on. Each site has an ID, which must match up to the ID of the Essentials Site itself, which in turn matches up with the name of the directory inside the Sites folder in Geocortex Essentials REST Elements. An example configuration for a Site is as follows:

<Sites>

<Site>

<ID>York</ID>

<ResourceDependencies>

<ResourceDependency>..\Workflows\HelloWorldWorkflow.xaml</ResourceDependency>

<ResourceDependency>..\..\REST\bin\Geocortex.Projects.YorkRegion.Workflows.dll</ResourceDependency>

<ResourceDependency>..\PrintTemplates</ResourceDependency>

<ResourceDependency>{DriveRoot}\c$\inetpub\wwwroot\SilverlightViewer\_1\_4\ClientBin\Resources\Images\SplashTitle.png</ResourceDependency>

<ResourceDependency>{DriveRoot}\c$\inetpub\wwwroot\SilverlightViewer\_1\_4\Config\Viewer.Settings.xml</ResourceDependency>

</ResourceDependencies>

</Site>

</Sites>

Each individual ResourceDependency will be copied to the specified location during site migration. A ResourceDependency is typically resolved relative to the site directory itself. If a fully-qualified location is required, it must begin with {DriveRoot}. During site migration, the location of the dependent resource will be resolved using the configuration of the DriveRoot from the appropriate EssentialsEnvironment.

Let’s consider an example. Assume there are two EssentialsEnvironments configured, one with a DriveRoot of [\\devserver](file:///\\devserver) and another with a DriveRoot of [\\testserver](file:///\\testserver). During migration of the York site as configured above from dev to test, the Resource Dependency of SplashTitle.png will be copied:

From:

[\\devserver\c$\inetpub\wwwroot\SilverlightViewer\_1\_4\ClientBin\Resources\Images\SplashTitle.png](file:///\\devserver\c$\inetpub\wwwroot\SilverlightViewer_1_4\ClientBin\Resources\Images\SplashTitle.png)

To:

[\\testserver\c$\inetpub\wwwroot\SilverlightViewer\_1\_4\ClientBin\Resources\Images\SplashTitle.png](file:///\\testserver\c$\inetpub\wwwroot\SilverlightViewer_1_4\ClientBin\Resources\Images\SplashTitle.png)

DependentResources may refer to individual files or entire directories. If a DependentResource directory is configured, the directory and all subdirectories will be migrated along with the site.

If a site has no dependent resources (as in, all of the components required for the site to function are contained inside the primary site directory in the REST Elements\Sites folder), then it is not necessary to configure the Site in the configuration XML. The ID of the site supplied to the SiteMigrationApp.exe program will simply refer to a directory inside the main Essentials Sites folder for the respective environments and will be copied without any resource dependencies. Note that token replacement will still occur.

### Backup Directory Organization

The backup directory contains a single directory for each site migration performed into a particular environment. No backups are made in the source environment, only in the destination environment during a migration.

The backup directory for a site is named using the following pattern:

yyyy-MM-dd HH mm ss ffff – SITEID

* yyyy: 4-digit year
* MM: 2-digit month
* dd: 2-digit day of month
* HH: 2-digit hour of day (24-hour clock)
* mm: 2-digit minute of hour
* ss: 2-digit second of minute
* ffff: 4-digit ten-thousandth of a second
* SITEID: ID of the site being migrated

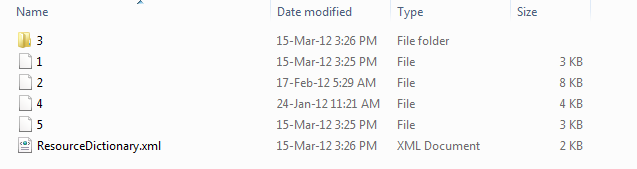
Example: 2012-03-15 15 25 38 2702 - York

Inside this directory will be a directory named for the ID of the site. If the site did not exist in the destination environment prior to the migration, no Site ID directory will be present in the backup folder.

Another folder will be present in the backup directory, containing the site’s dependent resources. This directory will always be present, even if it is empty. The name of the folder will be ResourceDependencies. Note that this organization precludes the migration of a Site with ID of ResourceDependencies (since the name of the ResourceDependencies folder must not match the name of the Site folder).

Inside the ResourceDependencies folder will be backups of all of the site’s dependent resources at the time of migration. If a dependent resource did not exist in the destination environment prior to migration, there will be no backup of it created (of course).

The dependent resources are named with a simple numbering scheme, starting at 1 and incrementing by 1 for each resource. Resource backups may be either files or folders – all adhere to the numbering scheme. The following is a screenshot of a site backup directory for the sample “York” site configured above.



There is an XML file located in the ResourceDependencies directory called ResourceDictionary.xml which contains simple number-filepath mappings for all of the dependent resources. This enables the site migration application to automatically restore backups to their original locations. This ResourceDictionary.xml file is automatically written to and read from during backup and restoration processes. It can be safely viewed, but editing the file should be done with caution as it could corrupt the backup references.