CentraSite Administrator's Guide

Version 9.7

October 2014

Document ID: CS-AG-97-20141015

Table of Contents

About this Guide

This guide describes the administration-level tasks that you can perform in the CentraSite environment.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Documentation Installation

You can download the product documentation using the Software AG Installer. The documentation is downloaded to a central directory named _documentation in the main installation directory (SoftwareAG by default).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products and certified samples, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

1 Basic Operations

Overview of Basic Operations

The CentraSite metadata repository consists of the following components:

- A registry for web services and related SOA (service oriented architecture) objects.
- A metadata repository.

This chapter describes how to perform administration-level CentraSite operations, such as the administration of runtime components of CentraSite, administration of the internal database that hosts the registry and repository, performance tuning and error analysis.

Notation

In this document, the following terms are used to describe the required disk locations:

Term	Description
<*SuiteInstallDir*>	This is the root directory for all products in the webMethods suite. On Windows, this is by default C:\SoftwareAG. On UNIX, this is by default /opt/softwareag/.
<*CentraSiteInstallDir*>	This is the CentraSite installation directory. By default, this is the CentraSite folder under <*SuiteInstallDir*> .
<*RuntimeDir*>	This is the location of the Software AG Runtime. By default: <*SuiteInstallDir*> /profiles/CTP.
<*RuntimeWebAppsDir*>	This is the directory where the web applications on the Software AG Runtime are deployed. By default: <*RuntimeDir*> /workspace/webapps.
<*JDKInstallDir*>	This is the installation directory of the Java JDK on Windows, for example <*SuiteInstallDir*> \jvm\jvm<*n*> , where <*n*> is the JDK version number.

Starting and Stopping the CentraSite Registry Repository

The CentraSite Registry Repository is started automatically when you boot your machine.

On Windows, the CentraSite Registry Repository runs as a Windows service. You can use the Windows Services application that is provided with the operating system to start and stop the service.

For other operating systems, and as an alternative method on Windows, you can start and stop the CentraSite Registry Repository from the command line.

Important: Before using the `inoadmin` tool on Windows or UNIX systems, it is important that you run the command line script `centrasite_setenv`. This ensures that environment variables and lookup paths are set correctly for the subsequent commands.

On Windows, the script is `centrasite_setenv.cmd`, and on UNIX the script is `centrasite_setenv.sh`. This script is located in <*SuiteInstallDir* >/CentraSite/bin.

Starting the CentraSite Registry Repository

Enter the following command on the command line:

```
inoadmin start CentraSite
```

Stopping the CentraSite Registry Repository

Use one of the following commands to stop the CentraSite Registry Repository:

Command	Description
inoadmin shutdown CentraSite	This option terminates the CentraSite Registry Repository session normally and waits for currently active processing to finish. The waiting time in seconds can be set with the server property Maximum Transaction Duration).
inoadmin cancel CentraSite	This option terminates the CentraSite Registry Repository immediately. User transactions that have not finished processing are rolled back.
inoadmin abort CentraSite	This option causes an emergency shutdown of the CentraSite Registry Repository. All processing is stopped immediately. Crash dump files will be written. This method will cause an automatic repair ("autorepair") the next time the CentraSite Registry Repository is started, and should only be used as a last resort.

You can also use Command Central to start and stop the CentraSite Registry Repository. The CentraSite Registry Repository is listed as a process in the instances list of your chosen Command Central environment, and you can use standard operations in the

Command Central interface to start and stop the process. Refer to the *Software AG Command Central Help* guide for details.

Administering the License Key

CentraSite is equipped with a license key that enables you to use the CentraSite software. The license key determines:

- Which edition of CentraSite you are licensed to use.
- The date until which your license is valid.

The following topics describes the purpose of the license key, and how to change the license key if required.

Relationship Between the License Key and Editions

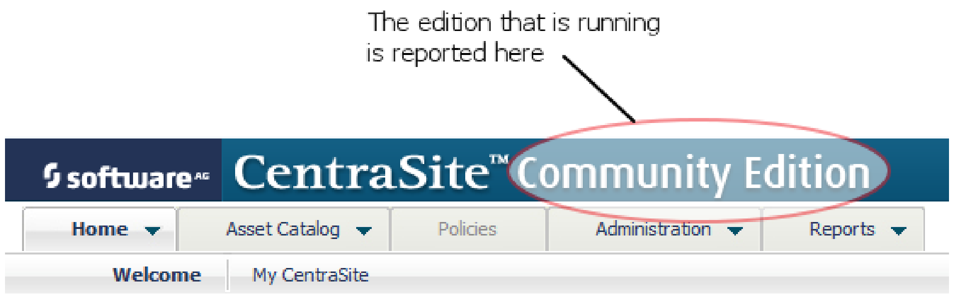
In addition to the full-feature CentraSite edition, Software AG also offers the free-of-charge CentraSite Community Edition. Your license key determines which edition is enabled for your instance of CentraSite.

If you do not specify a license key during the installation procedure, CentraSite is installed with a *default key*, which enables the Community Edition. The default key has no expiration date. If you are licensed to use the full-feature edition, you will receive an additional license key from Software AG, and you can specify this key either during the installation procedure or in a separate step after the installation procedure.

For information about the CentraSite editions and a list of specific features in each, see *Getting Started with CentraSite*.

How to Tell Which Edition of CentraSite You Are Using

To determine which edition of CentraSite is running, open CentraSite Control and examine the banner at the top of the screen. If the Community Edition is running, this is indicated in the banner, otherwise the full-feature edition is running.



Changing the License Key

You might wish to change the license key that CentraSite is using, for example in the following circumstances:

- Install the key that you have received from your software supplier.
- Replace an expired key.
- Switch to a different license key (e.g. to upgrade to a different edition of CentraSite).

To change a license key

1. Stop the CentraSite Registry Repository.
2. Identify the file system location where the file containing the current license key is stored. This is by default <*SuiteInstallDir* >/CentraSite/lkey.
3. Identify the file containing the current license key. This is by default inm<*nn*> .xml, where <*nn*> represents the product release number.

   Rename the current license key file to a name of your choice. Ensure that you keep a backup copy of this file, in case you wish to revert to this license key at a later stage.
4. Copy the file containing the new license key into this file system location. If the name of the new file is not the same as the name you were using so far for the license key file, rename the new file to the old file name.
5. Start the CentraSite Registry Repository.

Working with Time-Limited Licenses

Certain licenses have expiration dates. If you have a time-limited license, your instance of CentraSite will automatically revert to the Community Edition when the license expires.

You can check the expiration date by examining the contents of the license key file, as follows:

To check the expiration date of your license

1. Open the license key file in a text editor.
2. Locate the element `ExpirationDate`.
3. If this element contains the value `Unlimited`, there is no expiration date, i.e. the use of the license is unlimited.

   If this element contains a date, this date is the last date for which the license is valid.

Maintaining the CentraSite Internal Database

The contents of the CentraSite Registry Repository are stored physically in an internal database. The internal database typically exists as a set of files on disk for persistent storage, together with a large memory cache during normal runtime operation.

The workings of the internal database are not revealed to end users by means of any user interface or API. Only the product administrator can access and maintain the internal database.

Repository Monitoring

While the CentraSite Registry Repository is running, you can monitor the disk space and memory cache requirements of its internal database. Based on these values, you can optimize your installation for best use of memory and disk space.

Database Activity

To display the database activity information, use the following command:

```
inoadmin showactivity CentraSite
```

This displays the following items of information:

Item	Description
Bufferpool size	The total amount of memory available for caching database blocks.
Current used bufferpool size	The amount of memory currently being used to store cached database blocks.
Current number of Index blocks	The number of currently cached blocks from the Index part of the database.
Current number of Data blocks	The number of currently cached blocks from the Data part of the database.
Current number of Temp blocks	The number of currently cached blocks from the Temp part of the database.
Number of buffer flushes	The number of physical writes from the buffer pool to the disk that have occurred during the current CentraSite session.

Item	Description
	A buffer flush is the process whereby CentraSite copies the entire contents of the buffer pool to disk, then deletes the contents of the buffer pool. This frees up the buffer pool for subsequent logical I/O operations.
Logical reads	The number of database read operations that accessed the buffer pool during the current CentraSite session.
Physical reads	The number of database read operations that caused a physical disk access during the current CentraSite session.
Bufferpool hit rate	The ratio of times that a read operation was satisfied from the buffer pool rather than from the disk during the current CentraSite session, expressed as a percentage.
Current bufferpool hit rate	The buffer pool hit rate, limited to the time period between the previous and current activation of the command `inoadmin showactivity` CentraSite.
Flush limit	The maximum amount of buffer pool space that can be in use before a buffer flush takes place. If this value is exceeded, an automatic buffer flush occurs.
Modified pages in bufferpool	The ratio of modified pages to unmodified pages in the buffer pool, expressed as a percentage.
Dynamic pool size	The size of the dynamic pool. The dynamic pool is a separate cache area, not part of the main buffer pool, and is used as a work area for certain operations such as sort and search operations. The dynamic pool is shared across all users of the system.
Current used dynamic pool size	The amount of the dynamic pool currently in use.
Maximum pool usage	The high-water mark of the dynamic pool usage during the current session.
Current number of space waiters	The number of users/applications that are currently waiting for space allocation in the dynamic pool.

Item	Description
Total number of space waiters	The total number of users/applications that have waited for space allocation in the dynamic pool during the current session.

Space Usage

You can display information about the physical disk requirements of CentraSite's internal database. The database consists of several components called *spaces*, and each database space is stored in its own physical file. There are several kinds of database space:

Database space	Description
Data	Contains the data of the CentraSite Registry Repository. The file type of a data space is `2D0`. An example of a file name is `AAB00001.2D0`.
Index	Contains the indexes that CentraSite maintains for retrieving stored data. The file type of an index space is `2I0`. An example of a file name is `AAB00001.2I0`.
Journal	Contains information required for rolling back transactions. The file type of a journal space is `2J0`. An example of a file name is `AAB00001.2J0`.
Log	Contains a log of all database modifications that have occurred in the current CentraSite session. The file type of a log space is `2L0`. An example of a file name is `AAB000010000000052.2L0`. The name is composed of the filename of the database's data and index spaces (for example, `AAB00001`), followed by a sequence number. The sequence number is incremented by 1 for each new log space.
Backup	Contains a backup of the CentraSite Registry Repository. There is an initial backup The file type of a backup space is `2B0`. An example of a file name is `AAB00001001334723430.2B0`. The name is composed of the filename of the database's data and index spaces (for example, `AAB00001`), followed by an integer that represents the date and time when the backup was created.

To display information about the database spaces, use the following command:

```
inoadmin listdbspaces CentraSite
```

For each database space, the following information is displayed:

- The type of database space, e.g. Data, Index, Journal, Log or Backup. There can be several spaces of the same type.
- The amount of disk storage that this database space uses.
- The location of the database space in the file system and the name of the physical file that contains the database space.

Backing Up the Database

To protect against accidental loss of data, we recommend you to take regular backups of the internal database in which CentraSite's registry and repository data is stored. When you make a backup, you copy the contents of the internal database to a file on the file system. At a later stage, you can retrieve the contents of a backup and restore them into the internal database.

When you create a backup, the backup file is stored by default in the predefined location for backups, but you can optionally specify a different backup location. For more information about predefined locations, see "Locations" on page 31.

Note: During the installation of CentraSite, a backup of the initial database state is automatically created, with a timestamp equal to the date and time when you install the product. If you for any reason wish to restore the database to its initial state, i.e., the state immediately after product installation, you can use this backup.

To back up the internal database

1. Decide whether you want to create the backup in the default backup location or in a different location.
2. If you want to create the backup in the default backup location, use the following command:

   ```
   inoadmin backup CentraSite
   ```

   If you want to create the backup in a location other than the default backup location, use the following command:

   ```
   inoadmin backup CentraSite <Location>
   ```

   where <*Location*> is the path where the backup will be created, for example:

   ```
   inoadmin backup CentraSite C:\SoftwareAG\AnotherBackupLocation
   ```

   The location you specify must exist already, otherwise the backup will not run.

The Backup Key

When the backup completes, a status message is output on the command line, indicating the date and time when the backup was created. In addition, a backup key is displayed.

The backup key is a unique identifier for the backup. If you wish to restore the backup at a later stage, you identify the backup using the backup key.

Restoring the Database from a Backup

You can change the contents of CentraSite's internal database back to a previous state by restoring a backup. When you restore a backup, you completely replace the current contents of the database by the contents that existed when the backup was made.

Repository changes that are made between one backup and the subsequent backup are stored in session logs. When you restore from a backup, you can optionally choose to include or omit the data from the session logs.

As soon as the restore step successfully finishes, the repository is automatically started in standby mode. Then the recover step is started, in which all changes that were made since the last backup are reapplied from the session logs. Finally, the repository is shut down again. The restore function can only be used when the repository is inactive (stopped).

Note: During the installation of CentraSite, a backup of the initial repository state is automatically created, with a timestamp equal to the date and time when you install the product. If you for any reason wish to restore the repository to its initial state, i.e., the state immediately after product installation, you can use this backup.

To restore the database from a backup

1. First make a backup of the current database, in case you decide to return to this database state at a later time.
2. If the CentraSite Registry Repository is running, stop it before continuing.
3. Decide which backup file you wish to use for the restore operation. You can show a list of all available backups and their backup keys (the unique identifiers for the backups) by using the command:

   ```
   inoadmin listbackups CentraSite
   ```

4. Use a command of the following form to restore from the chosen backup:

   ```
   inoadmin restore CentraSite <BackupKey> <RecoverOption>
   ```

   Select the recover option that you want to use. If you do not specify a recover option, this has the same effect as using the option `recover all`. The following recover options are available:

Option	Action
recover all	The database will be restored to the state stored in the backup, and all session logs created since the backup are included. This is the default option.

Option	Action
recover no	The database will be restored to the state stored in the backup. No session log data will be processed.
recover <*UntilDateTime*>	All session logs created after the specified time and date are excluded from the recovery. The format of this field is: `DD-MMM-YYYY:HH:MM:SS` Note that the month is given as a 3-letter abbreviation, using the first 3 characters of the month's name. Example: 23-OCT-2012:16:52:30
recover	This has the same effect as `recover all`.

The following example restores the database from a backup whose backup key is `001334732430`. All database changes that occurred after the backup was made, up to and including 11:30am on October 25, 2012, will also be retrieved from the session logs.

```
inoadmin restore CentraSite 001334732430 recover 25-OCT-2012:11:30:00
```

Disabled Backups

If you choose to restore a backup without full recovery by using the recovery option `no`, and there are one or more backups that are more recent than the backup being restored, these more recent backups will be set to disabled.

If you choose to restore a backup without full recovery by using the option `recover` <*UntilDateTime*>, and there are one or more backups that are more recent than the time specified by this option, these more recent backups will be set to disabled.

A disabled backup cannot be accessed any more from any of the CentraSite user interfaces, and in particular cannot be accessed for restore or recover operations. It is not displayed in the list of available backups.

Disabled backups remain on the backup location on your disk as long as there are older, non-disabled backups. If you delete all non-disabled backups that are older than a given disabled backup, CentraSite automatically deletes the disabled backup.

If you should wish to reactivate a disabled backup, please archive it using standard operating system functionality and contact your software supplier.

Moving a database / Disaster Recovery

Under normal circumstances, a database backup that is created on one machine can only be restored to the same machine. However, a database backup that originated on one machine can be configured, so that it can be restored onto another machine. For more information, see "Moving a Database to Another Location" on page 30.

Deleting a Backup

You can delete backups that are no longer required. Deleting a backup removes all of the backup spaces that are associated with it, but the associated session log information is not removed, since it may subsequently be required if the database has to be recovered.

To delete a backup

1. Identify the backup that you wish to delete. You can list the available backups and their backup keys using the command:

   ```
   inoadmin listbackups CentraSite
   ```

2. Enter the following command:

   ```
   inoadmin deletebackup CentraSite <BackupKey>
   ```

   where <*BackupKey*> is the backup key of the backup you wish to delete.

   Example:

   ```
   inoadmin deletebackup CentraSite 001334732430
   ```

Note: If, after you delete a backup, there are one or more disabled backups older than the oldest remaining non-disabled backup, CentraSite automatically deletes all of these older disabled backups. For more information, see "Restoring the Database from a Backup" on page 28.

Moving a Database to Another Location

Under certain circumstances you might wish to move a CentraSite database from an existing CentraSite installation into another CentraSite installation, either on the same machine or on a different machine. Examples of such situations are:

- When resources for processing become insufficient.
- In a disaster recovery scenario.
- As part of a side-by-side product installation, whereby two versions of CentraSite can exist on the same machine.

Moving a CentraSite database to another machine is not supported in the following scenarios:

- if the architectures of the source and the target machine is with different byte orders (big-endian versus little-endian)
- if the source machine is a Unix system and the target machine is a Windows system.

If you are not sure whether moving a database is supported in your environment, contact a technical representative of Software AG.

The CentraSite registry/repository contains environment-dependent data. This data has to be adjusted to the new environment when the data is moved to another machine.

The CentraSite kit contains a command line script that allows you to create a new database from an existing backup and to modify the data to the needs of the new environment. On Windows, the script is `MoveCentraSiteRR.cmd`, and on UNIX the script is `MoveCentraSiteRR.sh`.

The steps performed by the script are:

- Calculate database space sizes
- Delete the existing CentraSite database
- Create a new CentraSite database
- Adjust the environment specific data
- Create a new backup containing the modifications

The script is called with 3 parameters as follows:

```
MoveCentraSiteRR.cmd  <BackupFilename> <Username> <Password>
```

The parameter <*BackupFilename*> is the name and path of the existing backup file. The script can only be run using the credentials of the predefined user `DefaultUser`, so <*Username*> must be `DefaultUser` and <*Password*> must be the password of this user.

The script must be called from the `bin\cfg` location of the target CentraSite installation.

Important: The contents of the backup file will overwrite the database of the target CentraSite installation.

Example:

```
cd C:\SoftwareAG\CentraSite\bin\cfg
MoveCentraSiteRR.cmd C:\temp\CS820test.2B0 DefaultUser PwdForDefUser
```

Locations

CentraSite uses certain default locations on disk to store information about the active CentraSite Registry Repository session and the backup environment. These locations are:

Location type	Purpose	Default path
temporary	The Temporary Working Location. This location contains temporary data that is required during normal database operation.	<*CentraSiteInstallDir*> \data
backup	The Backup Location. This is the location where backup files are created by default.	<*CentraSiteInstallDir*> \data

Location type	Purpose	Default path
log	The Log Location. This is the location where session log files are created by default. If the log archive location is defined, the log location holds only the most recent session log, and all other logs are stored in the log archive location.	*<CentraSiteInstallDir>* \data
archive	The Log Archive Location. This is the location where all session logs other than the current session log are stored. If no log archive location is defined, all session log files are held in the log location, regardless of whether they are the log file of the current session or a previous session.	(no default defined)
reserved	The Reserved Location. This location is used as an overflow area if the database's standard locations have filled up.	*<CentraSiteInstallDir>* \data

The CentraSite installation procedure defines default paths for each of these location types. Depending on your storage requirements, you can change these defaults to use different or additional paths.

Listing the Currently Defined Locations

You can list the currently defined locations as follows:

To list the currently defined locations

1. Use the following command:

```
inoadmin showlocations CentraSite
```

Changing the Currently Defined Path of a Location

You can change the currently defined path of a location as follows:

To change the currently defined path of a location

1. Use a command of the following form:

```
inoadmin setlocation CentraSite <LocationType> <path> [<path> ...]
```

where *<LocationType>* is one of the location types shown in the table above. The location can be assigned to more than one path. If several paths are defined, they are used in the order specified from left to right; when there is no more disk space available in a path, the next path is used.

If the path contains spaces or characters that the operating system treats as escape characters in the command line (for example, the backslash character \ in Windows), you must enclose the path in quotes.

The path or paths you specify completely replace the previously specified path or paths. If you want to extend the current path specification with an additional path, you need to use the `inoadmin setlocation` command and specify both the existing path and the new path.

The following example sets the new default path for the Backup location to C:\backuploc1 and D:\backuploc2. This means that C:\backuploc1 will be used as long as there is available disk space; if it is full, then D:\backuploc2 will be used instead.

```
inoadmin setlocation CentraSite backup C:\backuploc1 D:\backuploc2
```

Database Configuration Parameters

You can configure various properties of CentraSite's internal database. The available properties are:

Property	Description
buffer pool size	This defines the size of the buffer pool that is used for storing intermediate results during normal processing.
maximum transaction duration	This defines the maximum time, in seconds, that a transaction is allowed to exist.
non-activity timeout	This defines the maximum time, in seconds, that a transaction is allowed to be inactive.
XML work threads	This defines the number of XML-processing threads that be active concurrently in the internal processing engine.
XML maximum sessions	This defines the maximum number of user sessions that CentraSite can process concurrently.
number of backup generations	The number of full backups that CentraSite will keep in parallel. When this number is exceeded, the oldest backup and the corresponding log spaces will be deleted.

Property	Description
write_limit	The amount of the modified buffer pool space that triggers a flush (disk write) of the modifications present in the buffer pool. The default for the buffer pool size is 60MB. If this property is set to 0, CentraSite will adjust the flush limit automatically.

For each property, the following information is available:

- **handle**
- **minimum**

  The minimum allowed value that can be configured for this property.
- **maximum**

  The maximum allowed value that can be configured for this property
- **default**
- **configured**
- **current**

  The current value of the property.
- **type**

  The datatype of the property (string, numeric etc.).
- **unit**

  The unit of measurement for the property's value, e.g. megabytes or seconds.
- **state**

If you wish to examine or modify the database properties, you have the following possibilities:

Command	Description
`inoadmin listproperties CentraSite`	Show a list of all of the available database properties and their values. The information displayed includes (where appropriate) the property name, the maximum and minimum allowed values, the current value, the configured value, high water mark.
`inoadmin getproperty CentraSite <PropertyName>`	Show the value of the given property.

Command	Description
	If the property name contains one or more spaces, enclose the name in quotes.
`inoadmin setproperty CentraSite <PropertyName><PropertyValue>[no]restart`	Change the value of the given property to the given value. If the property name or property value contains one or more spaces, enclose the name or value in quotes. The changed value will take effect at the next restart of the CentraSite Registry Repository. You can cause an automatic restart by specifying `restart`, otherwise specify `norestart`.
`inoadmin setproperties CentraSite <XMLInputFile> [no]restart`	Change the values of several properties to the values specified in the supplied XML file. The XML file must use the same element structure as the XML file created by the `inoadmin listproperties` command. The changed values will take effect at the next restart of the CentraSite Registry Repository. You can cause an automatic restart by specifying `restart`, otherwise specify `norestart`.

Reclaiming Disk Space in the CentraSite Database

Within the CentraSite internal database a considerable amount of data may be stored temporarily, for example, log data that is purged regularly or metrics data stored by other webMethods sub systems. When such data is stored the space required for the database increases. However, after deleting such data the space allocated will remain. Although the allocated space is re-used when additional data of the same type is stored again, the administrator may prefer to reorganize the database so that unused space can be reclaimed. To reclaim space in the CentraSite internal database, use the `inoadmin reorganize` command. The `reorganize` command reduces the disk space by returning blocks no longer used to the file system. The organize command can be used only on an inactive database. Attempting to reorganize an active database is not allowed.

How to Reclaim Disk Space in the CentraSite Database

If you want to reclaim disk space in the CentraSite database, proceed as follows:

To reclaim database space

1. Stop the database you want to reorganize.
2. Ensure there is enough space on the database backup location to hold one full backup of the CentraSite database. This space is required because the `reorganize` command creates a temporary backup while it defragments the database. The temporary backup will be deleted when the reorganize command completes successfully.
3. To check the available free disk space, execute the following command:

   ```
   inoadmin reorganize CentraSite evaluate
   ```

   Output from the evaluate command shows the current amount of free disk space.

   For example:

   ```
   INODSI1183: 161.84 MB empty data space found.
   ```

   The reorganize function uses a temporary backup to defragment the free space. Thus it is a pre-requisite that the space required for one CentraSite backup is available on the backup location. This temporary backup will be deleted when the reorganize function completes successfully.
4. To start database reorganization, use the following command:

   ```
   inoadmin reorganize CentraSite
   ```
5. If the `reorganize` command completes successfully, restart the database.

Configuring the Authentication Settings

The authentication in the CentraSite Registry Repository is configured with default settings during installation. You can define additional authentication configurations, and you can change the default configuration to be one of the additional configurations.

The default authentication configuration determines the user repository that will be used to authenticate users who log on to CentraSite. Initially, the default user repository is CentraSite's own user repository, which has the domain name INTERNAL. You might want to define additional configurations that define for example an LDAP user repository.

You can view and modify the authentication settings using the command line tool `CentraSiteCommand.cmd` (Windows) or `CentraSiteCommand.sh` (UNIX). The command line tool is located in <*SuiteInstallDir*>/CentraSite/utilities.

If you start the command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

For more information, see "Creating Authentication Configurations" on page 66.

Listing Names of All of the Configurations

To list the names of all of the currently defined authentication configurations, use a command of the following format:

```
CentraSiteCommand list Authentication
```

Input Parameters

None.

Note: The displayed list shows also which configuration is the default.

Listing Details of a Particular Configuration

To list details of a particular configuration, use a command of the following format:

```
CentraSiteCommand get Authentication -domain <DOMAIN>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `get Authentication` utility:

Parameter	Description
`-domain`	The domain name of the user repository associated with the configuration.

For example:

```
CentraSiteCommand get Authentication -domain LDAPDomain
```

The details are returned as an XML file. The XML file has a root element `ino:domain` that has the following attributes:

Element name	Description
`ino:acceptusers`	Meaning: This specifies whether to allow access of any user that is correctly authenticated by the authentication service or whether to only allow access by users that are explicitly defined in CentraSite. Possible values: `all` - Allow access of any user that is correctly authenticated by the authentication service; `defined` (default value) - allow access only to users defined in CentraSite

Element name	Description
`ino:casesensitiveuserids`	Meaning: This determines whether or not user names in this domain are case-sensitive. Possible values: `true` - user IDs in this domain are case-sensitive; `false` - user IDs are not case-sensitive
`ino:default`	Meaning: This determines whether or not the configuration is the default configuration. Possible values: `true` - This is the default configuration; `false` - this is not the default configuration.
`ino:domainid`	Meaning: The domain name of the user repository associated with the configuration.
`ino:domaintype`	Meaning: The type of user repository associated with the configuration. Possible values: Typical values are: `INTERNAL` (the default domain), or a Windows domain name or an LDAP domain name.
`ino:expire`	Meaning: The amount of time (in seconds) that the user is cached in the server after successful authentication. Changes made to the user, e.g. deletion or password changes, do not take effect until this time has elapsed. The default is 120 seconds. This setting is provided for performance reasons. A value of 120 seconds is reasonable. If the connection to the LDAP server is slow, you can increase this figure.
`ino:usegroups`	Meaning: This specifies whether to use the external group information from domains; for example, the groups in an Active Directory Server or in an LDAP server. Possible values: `true` - use external group information; `false` (default value) - do not use external group information.

Example

Here is an example of an authentication configuration returned as an XML file:

```
<ino:domain xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
  ino:acceptusers="all" ino:casesensitiveuserids="false" ino:default="false"
  ino:domainid="LDAP" ino:domaintype="ldap" ino:expire="120"
  ino:usegroups="true">
    <ino:param ino:content="ldap://ldapserver12" ino:name="host"/>
    <ino:param ino:content="10389" ino:name="port"/>
    <ino:param ino:content="ApacheDS" ino:name="ldap_server_type"/>
    <ino:param ino:content="ou=people,ou=RegionNorth,o=WidgetCo"
       ino:name="ldap_person_dn"/>
    <ino:param ino:content="inetOrgPerson" ino:name="ldap_person_object"/>
    <ino:param ino:content="cn" ino:name="ldap_user_field"/>
    <ino:param ino:content="ou=groups,ou=RegionNorth,o=WidgetCo"
       ino:name="ldap_group_dn"/>
    <ino:param ino:content="groupOfUniqueNames" ino:name="ldap_group_object"/>
    <ino:param ino:content="uniqueMember"
       ino:name="ldap_group_person_attribute"/>
    <ino:param ino:content="rd" ino:name="ldap_resolve_groups"/>
    <ino:param ino:content="TRUE" ino:name="useLdapTechUser" />
    <ino:param ino:content="c:\softwareag\centrasite\bin\cred.txt"
       ino:name="techLdapUserCredFile" />
    <ino:param ino:content="c:\softwareag\centrasite\bin\key.txt"
      ino:name="techLdapUserKeyFile" />
    <ino:configuration>
       <ino:group>
          <ino:properties>
              <ino:mapping ino:external="description" ino:local="description"/>
          </ino:properties>
       </ino:group>
       <ino:user>
          <ino:properties>
             <ino:mapping ino:local="organization" ino:external="org"/>
             <ino:mapping ino:local="emailAddresses:emailAddress:address"
                ino:external="mail"/>
             <ino:mapping ino:local="telephoneNumbers:telephoneNumber:number"
                ino:external="telephoneNumber"/>
             <ino:mapping
                ino:local="telephoneNumbers:telephoneNumber:countryCode"
                ino:external="telephoneCode"/>
              <ino:mapping ino:local="telephoneNumbers:telephoneNumber:extension"
                ino:external="telephoneExt"/>
             <ino:mapping ino:local="telephoneNumbers:telephoneNumber:areaCode"
                ino:external="telephoneAreaCode"/>
             <ino:mapping ino:local="personName:firstName" ino:external="cn"/>
             <ino:mapping ino:local="description" ino:external="description"/>
             <ino:mapping ino:local="postalAddresses:postalAddress:postalCode:"
                ino:external="postalcode"/>
             <ino:mapping ino:local="postalAddresses:postalAddress:city:"
                ino:external="postalcity"/>
             <ino:mapping
                ino:local="postalAddresses:postalAddress:stateOrProvince"
                ino:external="stateorprovince"/>
             <ino:mapping ino:local="postalAddresses:postalAddress:country"
                  ino:external="countrycode"/>
             <ino:mapping ino:local="URL" ino:external="E-mail"/>
          </ino:properties>
       </ino:user>
    </ino:configuration>
</ino:domain>
```

Setting the Default Configuration

To set the default configuration, use a command of the following format:

```
CentraSiteCommand set DefaultDomain -domain <DOMAIN>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set DefaultDomain` utility:

Parameter	Description
`-domain`	The domain name of the user repository associated with the configuration.

Example

```
CentraSiteCommand set DefaultDomain -domain LDAPdomain
```

An authentication configuration containing the specified domain must already exist in CentraSite.

Adding a Configuration

To add a configuration using a LDAP domain name, use a command of the following format:

```
CentraSiteCommand set Authentication -domain <DOMAIN>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set Authentication` utility:

Parameter	Description
`-domain`	The domain name of the user repository associated with the configuration.

Example

```
CentraSiteCommand set Authentication -domain LDAPdomain
```

Modifying a Configuration

To modify a configuration, use a command of the following format:

```
CentraSiteCommand set Authentication -domain <DOMAIN>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set Authentication` utility:

Parameter	Description
`-domain`	The domain name of the user repository associated with the configuration.

Example

```
CentraSiteCommand set Authentication -domain LDAPdomain
```

Removing a Configuration

To remove a configuration, use a command of the following format:

```
CentraSiteCommand remove Authentication -domain <DOMAIN>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `remove Authentication` utility:

Parameter	Description
`-domain`	The domain name of the user repository associated with the configuration.

Example

```
CentraSiteCommand remove Authentication -domain LDAPdomain
```

You cannot remove the pre-installed domain `INTERNAL`.

You also cannot remove a configuration that is the current default configuration. If you want to delete such a configuration, you must first change the default configuration to another configuration.

Configuring Port Numbers

This topic gives information about HTTP/HTTPS port numbers used by CentraSite components. In general there is no need to change these values, unless your site's requirements differ from the CentraSite default values.

Changing the Port Numbers of the Registry/Repository

The CentraSite Registry Repository uses several port numbers that have the following default values:

Port Name	Description	Default port number
XML XTS port	The application port.	53301
ADMIN port	The administration port. The port used for access to the CentraSite Registry Repository server for all administration actions.	53303
HTTP port	The HTTP Server port. The TCP/IP port used for HTTP access to the CentraSite Registry Repository.	53313

If you wish to change these port numbers, the following locations must be updated:

- The registry of the machine on which the CentraSite Registry Repository is installed.
- The CAST web applications (only if the HTTP port is updated).

Important: To avoid inconsistencies, it is important to modify the port numbers in all locations in a single step.

Changing a Port Number on the Registry/Repository Host

The procedure to change a configured CRR port on the Registry/Repository host is as follows:

To change a configured CRR port on the Registry/Repository host

1. Run the command line script `centrasite_setenv`. This script is located in <*SuiteInstallDir* >/CentraSite/bin.

   This ensures that environment variables and lookup paths are set correctly for the following steps.

2. Check the current value of the port number by using a command of the following form at the operating system command prompt:

   ```
   inoadmin getproperty CentraSite <ParameterName>
   ```

   The <*ParameterName*> can be any of the port names shown in the above table. Example:

   ```
   inoadmin getproperty CentraSite "HTTP port"
   ```

The inoadmin program is available under the location <*SuiteInstallDir* >/CentraSite/bin.

3. Assign a new port number by using the following command at the operating system command prompt:

   ```
   inoadmin setproperty CentraSite <ParameterName> <NewPortNumber> norestart
   ```

   where <*NewPortNumber*> is the new port number that you wish to use.

4. Stop the CentraSite Registry Repository and start it again.

Changing the Software AG Runtime Port Numbers

The default Software AG Runtime port numbers for the CAST components are 53307 for plain HTTP communication and 53308 for HTTPS communication. The port 53305 is used to provide compatibility with the previous product releases as it was used by the CRR.

The port numbers are configured in property files that are located in <*SuiteInstallDir*> / profiles/CTP/configuration/com.softwareag.platform.config.propsloader. If for any reason these port numbers are unsuitable (for example, your environment might require these port numbers for a non-Software AG application), you can change them in the appropriate property files in this location as follows:

- `com.softwareag.catalina.connector.http.pid-CentraSite.properties`

  This file contains parameter settings for HTTP communication.

- `com.softwareag.catalina.connector.https.pid-CentraSite.properties`

  This file contains parameter settings for HTTPS communication.

- `com.softwareag.catalina.connector.http.pid-CentraSite-CrrHttp.properties`

  This file is provided for compatibility with previous product releases, for which different default port numbers were used.

In general, Software AG Runtime has to be restarted for the changes to take effect.

Configuring Secure Communication Between Components

This topic gives information about how to set up secure communication between CentraSite components, on the basis of SSL.

If you change the default configuration, you might also need to modify other products based on CentraSite. Changing the CAST configuration can affect applications such as:

- Clients that use the CAST web applications.
- Web services deployed by CentraSite-enabled products.

Secure Communication Between the CRR and the CAST

The communication between the CRR and the CAST components takes place via 2-way SSL authentication. For this full client/server SSL communication, the client and server must accept each other's certificates. This means that the CAST and CRR stores need to have matching certificates for the communication to work.

The CAST components have access to an SSL context to establish an SSL (HTTPS) connection to the CRR. The SSL authentication establishes a trusted relationship between the CentraSite Server on the CAST and the CRR. Therefore no user re-authentication needs to be performed by the CRR.

The CentraSite installation comes with self-signed certificates from Software AG.

You can configure a secure communication between the CRR and CAST by executing the following commands in the command line interface `CentraSiteCommand.cmd` (Windows) or `CentraSiteCommand.sh` (UNIX) of Command Central. The command line tool is located in <*SuiteInstallDir* >/CentraSite/utilities.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-file` must be specified as shown and not as `-FILE`.

Note: Keep in mind that you must execute the `AST` or `RR` command on the machine hosting an CAST or CRR environment.

You can disable the SSL communication between the CRR and the CAST components. However, Software AG strongly recommends you not to do this, because it opens a potential security risk.

Setting the Security Configuration for the Registry Repository

To define the SSL security values for use in the CentraSite Registry Repository environment from the command line, you must perform the following high-level steps:

- Create a script (RR-config.xml) file.
- Execute the configuration file with appropriate input parameters.

Creating a RR Configuration File to Define Security Values

Create a RR configuration file to define the SSL security values specific for Application Server Tier environment. The configuration RR-config.xml file should look as follows. Examine the RR-config.xml file. It contains at least the XML namespace used for providing uniquely named elements and attributes.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
  <properties>
    <entry key="com.softwareag.centrasite.security.keyStore">
      C:/SoftwareAG/CentraSite/test/files/certs/castcert.p12</entry>
    <entry key="com.softwareag.centrasite.security.keyStorePassword">
      cscert</entry>
    <entry key="com.softwareag.centrasite.security.keyStoreType">PKCS12</entry>
    <entry key="com.softwareag.centrasite.security.trustStore">
      C:/SoftwareAG/CentraSite/test/files/certs/casttrust.p12</entry>
    <entry key="com.softwareag.centrasite.security.trustStorePassword">
      cscert</entry>
    <entry key="com.softwareag.centrasite.security.trustStoreType">
      PKCS12</entry>
    <entry key="com.softwareag.centrasite.security.crr.trustStore">
      C:/SoftwareAG/CentraSite/test/files/certs/crrtrust.pem</entry>
    <entry key="com.softwareag.centrasite.security.crr.certificate">
      C:/SoftwareAG/CentraSite/test/files/certs/crrcert.crt</entry>
    <entry key="com.softwareag.centrasite.security.crr.keyFile">
      C:/SoftwareAG/CentraSite/test/files/certs/crr.key</entry>
    <entry key="com.softwareag.centrasite.security.crr.storePassword">
      cscert</entry>
  </properties>
```

The key and certificate files need to be in an OpenSSL readable format. The CA file needs to be in PEM format.

Note that in the default configuration, the same CA certificate is used for both client and server certificates.

The server parameters can be changed via the command line tool inoadmin.

The general syntax is

```
inoadmin setproperty CentraSite <PropertyName> <PropertyValue> norestart
```

For example:

```
inoadmin setproperty CentraSite "SSL certificate file"
"C:/SoftwareAG/CentraSite/files/certs/custom_cacert.pem" norestart
```

Executing the RR Configuration File That Invokes Security Values

To define the SSL security values for CRR, run your RR configuration file with the following required input parameters:

```
CentraSiteCommand set SSL RR -url <CENTRASITE-URL> -user <USER-ID>
-password <PASSWORD> -file <CONFIG-FILE>
```

Note: If you change the default configuration, this command will modify the SSL configuration for RR. A time stamped archive of the previous configuration will be available in the configuration file cast-config.YYYY-MM-DD_HH-MM-SS.xml in the folder <*SuiteInstallDir*> /CentraSite/cfg/archive.

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set SSL RR` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file, RR-config.xml, containing the security properties. If relative, the path should be relative to the location from where the command is executed.

Example

```
CentraSiteCommand set SSL AST -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -file RR-config.xml
```

Setting the Security Configuration for the CAST Components

To define the SSL security values for use in the CentraSite Application Server Tier environment from the command line, you must perform the following high-level steps:

- Create a script (AST-config.xml) file.
- Execute the configuration file with appropriate input parameters.

Creating a AST Configuration File to Define Security Values

Create a AST configuration file to define the SSL security values specific for Application Server Tier environment. The configuration AST-config.xml file should look as follows. Examine the AST-config.xml file. It contains at least the XML namespace used for providing uniquely named elements and attributes.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
  <properties>
    <entry key="com.softwareag.centrasite.security.keyStore">
      C:/SoftwareAG/CentraSite/test/files/certs/castcert.p12</entry>
    <entry key="com.softwareag.centrasite.security.keyStorePassword">
      cscert</entry>
    <entry key="com.softwareag.centrasite.security.keyStoreType">
      PKCS12</entry>
    <entry key="com.softwareag.centrasite.security.trustStore">
      C:/SoftwareAG/CentraSite/test/files/certs/casttrust.p12</entry>
    <entry key="com.softwareag.centrasite.security.trustStorePassword">
cscert</entry>
    <entry key="com.softwareag.centrasite.security.trustStoreType">
      PKCS12</entry>
  </properties>
```

Note: When AST and RR components are authenticated with the 2-way SSL environment, the authentication will not work if the security configuration of one of the components AST or RR is modified. So if you intend to modify the default security configuration, ensure that you modify the configuration for both components AST and CRR. In addition, make sure that you execute the `set SSL RR` command before you execute the `set SSL AST` command.

Executing the AST Configuration File That Invokes Security Values

To define the SSL security values for CAST, use a command of the following format:

```
CentraSiteCommand set SSL AST -url <CENTRASITE-URL> -user <USER-ID>
-password <PASSWORD> -file <CONFIG-FILE>
```

Note: If you change the default configuration, this command will modify the SSL configuration for other web applications that communicate with the RR. A time stamped archive of the previous configuration will be available in the configuration file cast-config.YYYY-MM-DD_HH-MM-SS.xml in the folder <*SuiteInstallDir*> /CentraSite/cfg/archive.

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set SSL AST` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file, AST-config.xml, containing the security properties. If relative, the path should be relative to the location from where the command is executed.

Example

```
CentraSiteCommand set SSL AST -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -file AST-config.xml
```

Obtaining the Security Configuration of Registry Repository

To fetch the SSL security values used in the CentraSite Registry Repository environment, use a command of the following format:

```
CentraSiteCommand get SSL RR -file <CONFIG-FILE>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `get SSL RR` utility:

Parameter	Description
`-file`	The absolute or relative path to the XML configuration file, RR-config.xml, containing the security properties. If relative, the path should be relative to the location from where the command is executed.

Example

```
CentraSiteCommand get SSL RR -file RR-config.xml
```

Obtaining the Security Configuration of CAST Components

To fetch the SSL security values used in the CentraSite Application Server Tier environment, use a command of the following format:

```
CentraSiteCommand get SSL AST -file <CONFIG-FILE>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `get SSL AST` utility:

Parameter	Description
`-file`	The absolute or relative path to the XML configuration file, AST-config.xml, containing the security properties. If relative, the path should be relative to the location from where the command is executed.

Example

```
CentraSiteCommand get SSL AST -file AST-config.xml
```

CAST Stores

The CentraSite installation comes with self-signed certificates from Software AG. These are:

- The keystore certificate, which is located in <*SuiteInstallDir*>/CentraSite/cast/files/certs/castcert.p12. It contains the client certificate and private client key.
- The truststore certificate, which is located in <*SuiteInstallDir*>/CentraSite/cast/files/certs/casttrust.p12. This would normally contain the server certificate but actually contains the CA certificate and key.

These files need to be in a Java readable format.

Note that in the default configuration, the same CA certificate is used for both client and server certificates.

Identifying the Communication Method Between CAST and CRR

To identify the communication method between the CentraSite Application Server Tier and Registry Repository, use a command of the following format:

```
CentraSiteCommand get SSL usage
```

Input Parameters

None.

Note: The command output shows "https" if SSL usage is enabled; and "http" if disabled.

Allowing HTTP Communication Between CAST and CRR

It is possible to change the communication between CAST and CRR from full 2-way SSL (HTTPS) communication to HTTP communication.

Caution: Software AG strongly advises you to use 2-way SSL at all times for this communication. If you intend to use HTTP rather than HTTPS communication, please consider carefully that using HTTP communication raises a potential security risk.

Some internal communication between CAST and CRR must always use SSL, therefore you cannot switch off HTTPS altogether.

Before you change the SSL communication between CAST and CRR, make sure you use `inoadmin` to change the communication method setting as follows:

```
inoadmin setproperty CentraSite "communication method" "HTTP and HTTPS" restart
```

The `inoadmin` program is available under the location <*SuiteInstallDir*>/CentraSite/bin.

To change the SSL communication between CAST and CRR, use a command of the following format:

```
CentraSiteCommand set SSL usage -user <USER-ID> -password <PASSWORD>
-value <http/https>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set SSL usage` utility:

Parameter	Description
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-value`	The values of either "http" or "https" to allow HTTP/HTTPS communication between AST and RR.

Example

```
CentraSiteCommand set SSL usage -user Administrator -password manage
-value http
```

Secure Communication Between Software AG Runtime and External Clients

Overview

In the CentraSite environment, Software AG Runtime can receive requests from clients such as:

- User applications using an API to communicate with the registry repository.
- Components of the Software AG Designer.

By default, only basic communication encryption without authentication is configured.

For information on how to configure SSL-based authentication and protect Tomcat, see Tomcat 7.0 documentation and product information at http://tomcat.apache.org/.

Software AG Runtime Properties File for SSL Communication

The file `com.softwareag.catalina.connector.https.pid-CentraSite.properties` located in <*SuiteInstallDir*> /profiles/CTP/configuration/com.softwareag.platform.config.propsloader contains the properties that you need to set in order to configure Software AG Runtime for secure communication with external clients. The properties in this file define the SSL keystore and SSL truststore that Software AG Runtime will use.

Runtime properties for SSL communication in the *Working with Software AG Runtime*. HTTPS connectors to set up the SSL environment. Note that this cross-product document refers to the properties file generically as `com.softwareag.catalina.connector.https.pid-<port_number>.properties`.

SSL Keystore

CentraSite comes with a sample keystore that contains self-signed certificates which are located in <*SuiteInstallDir*> /profiles/CTP/configuration/tomcat/conf and need be replaced if SSL-based authentication is to be used.

Please acquire and provide your own server certificate and define its location with the parameter `keystoreFile` (replace the default value) in the Software AG Runtime properties file for SSL communication.

Note that the CN of the certificate needs to be identical to the URL the server is addressed under, without the "https://". For example, for a server reachable under `https://MyWebServer:8443/`, the CN needs to be `MyWebServer`. Software AG Runtime supports both Java keystores (keystoreType="JKS", which is the default), and PKCS#12 keystores (keystoreType="PKCS12"). Please set the keystore password accordingly (parameter `keystorePass` in the Software AG Runtime properties file).

SSL Truststore

If you want to use client authentication for 2-way SSL, you need to set clientAuth="true" in the Software AG Runtime properties file for SSL communication, and supply a truststore, which is a keystore containing the certificate chain and trust root for the client certificates for which you want to allow access.

In the properties file, you also need to provide the following properties:

- truststoreFile: the name and path of the truststore file
- truststorePass: the password for accessing the truststore
- truststoreType: the type of the truststore
- truststoreProvider: the provider of the truststore

Note on SSL Port Number

If a URL addresses a location using SSL, the URL must explicitly specify the port number of the location, even if the default port number for SSL (443) is to be used.

SSL Keystore

CentraSite comes with a sample keystore that contains self-signed certificates which are located in <*SuiteInstallDir*> /profiles/CTP/configuration/tomcat/conf and need be replaced if SSL-based authentication is to be used.

Please acquire and provide your own server certificate and define its location with the parameter `keystoreFile` (replace the default value) in the Software AG Runtime properties file for SSL communication.

Note that the CN of the certificate needs to be identical to the URL the server is addressed under, without the "https://". For example, for a server reachable under `https://MyWebServer:8443/`, the CN needs to be `MyWebServer`. Software AG Runtime

supports both Java keystores (keystoreType="JKS", which is the default), and PKCS#12 keystores (keystoreType="PKCS12"). Please set the keystore password accordingly (parameter `keystorePass` in the Software AG Runtime properties file).

SSL Truststore

If you want to use client authentication for 2-way SSL, you need to set clientAuth="true" in the Software AG Runtime properties file for SSL communication, and supply a truststore, which is a keystore containing the certificate chain and trust root for the client certificates for which you want to allow access.

In the properties file, you also need to provide the following properties:

- truststoreFile: the name and path of the truststore file
- truststorePass: the password for accessing the truststore
- truststoreType: the type of the truststore
- truststoreProvider: the provider of the truststore

Note on SSL Port Number

If a URL addresses a location using SSL, the URL must explicitly specify the port number of the location, even if the default port number for SSL (443) is to be used.

CentraSite comes with a sample keystore that contains self-signed certificates which are located in <*SuiteInstallDir*> /profiles/CTP/configuration/tomcat/conf and need be replaced if SSL-based authentication is to be used.

Please acquire and provide your own server certificate and define its location with the parameter `keystoreFile` (replace the default value) in the Software AG Runtime properties file for SSL communication.

Note that the CN of the certificate needs to be identical to the URL the server is addressed under, without the "https://". For example, for a server reachable under `https://MyWebServer:8443/`, the CN needs to be `MyWebServer`. Software AG Runtime supports both Java keystores (keystoreType="JKS", which is the default), and PKCS#12 keystores (keystoreType="PKCS12"). Please set the keystore password accordingly (parameter `keystorePass` in the Software AG Runtime properties file).

SSL Truststore

If you want to use client authentication for 2-way SSL, you need to set clientAuth="true" in the Software AG Runtime properties file for SSL communication, and supply a truststore, which is a keystore containing the certificate chain and trust root for the client certificates for which you want to allow access.

In the properties file, you also need to provide the following properties:

- truststoreFile: the name and path of the truststore file
- truststorePass: the password for accessing the truststore
- truststoreType: the type of the truststore

- truststoreProvider: the provider of the truststore

Note on SSL Port Number

If a URL addresses a location using SSL, the URL must explicitly specify the port number of the location, even if the default port number for SSL (443) is to be used.

Configuring the Registry Cache Settings

For performance reasons, CentraSite uses an internal cache to store registry objects accessed via JAXR. In some cases, you might want to modify the cache configuration settings, in order to determine the optimal setup for your system.

This command line tool allows you to display and to modify the JAXR-based configuration settings. The JAXR-based configuration settings apply to each connection.

The tool consists of an executable jar file CentraSiteCacheConfiguration.jar, which is located in <*SuiteInstallDir* >/CentraSite/bin.

Prerequisites

To be able to use the command line tool, please note the following points:

- The user specified in the command line must have the CentraSite Administrator role.
- The CentraSite Registry Repository must be online.
- The tool requires a Java 6 runtime.

Displaying the Cache Configuration

In order to display the cache configuration, run the tool with the DISPLAY keyword.

To display the cache settings for JAXR

- Use a command of the following form:

  ```
  java -jar CentraSiteCacheConfiguration.jar
  <CentraSiteURL> <AdministratorID> <password>
       DISPLAY
  ```

  for example:

  ```
  java -jar CentraSiteCacheConfiguration.jar
  "http://localhost:53307/CentraSite/CentraSite"
  DOMAIN\admin pAsSw0rD DISPLAY
  ```

Modifying the Cache Configuration

The SET keyword is followed by pairs of option and value. After the modification operation is completed, the tool will display the modified cache configuration. The following options may be specified:

- maxElementsOnHeap
- maxElementsOffHeap
- memoryStoreEvictionPolicy
- statistics

If one of the options is not specified in the SET operation, its existing value will be copied.

The SET keyword is used as follows:

To modify the cache settings for JAXR

- Use a command of the following form:

  ```
  java -jar CentraSiteCacheConfiguration.jar
  <CentraSiteURL> <AdministratorID> <password>
  SET <option> <value> [<option> <value> ...]
  ```

  for example:

  ```
  java -jar CentraSiteCacheConfiguration.jar
  http://localhost:53307/CentraSite/CentraSite
  DOMAIN\admin pAsSw0rD SET maxElementsOnHeap 10000
  ```

Option: maxElementsOnHeap

The option `maxElementsOnHeap` defines the maximum number of elements in the cache. A value of 0 means no limit. Once the cache is full, an element will be evicted according to the algorithm specified by the `memoryStoreEvictionPolicy` option.

Option: maxElementsOffHeap

Note: The use of this option requires a special license key. For further information contact your software supplier.

The option `maxElementsOffHeap` defines the amount of off-heap memory available to the cache. Off-heap memory is a separate unit of memory available outside of the conventional JVM heap which can be used for caching.

This option's values are given as `<number>k|K|m|M|g|G|t|T`, where the units can be kilobytes (k|K), megabytes (m|M), gigabytes (g|G) or terabytes (t|T).

For example, `maxMemoryOffHeap="2g"` allots 2 gigabytes to off-heap memory. A value of 0 means no off-heap memory.

Before using off-heap memory, direct memory space, also called direct (memory) buffers, must be allocated. In most popular JVMs, direct memory space is allocated using the Java property `-XX:MaxDirectMemorySize`. This memory space, which is part of the Java process heap, is separate from the object heap allocated by `-Xmx`. The value allocated by `-XX:MaxDirectMemorySize` must not exceed the physical RAM, and is likely to be less than the total available RAM due to other memory requirements. The value allocated to direct memory should be at least 32MB more than the off-heap memory allocated to the caches.

Option: memoryStoreEvictionPolicy

The option `memoryStoreEvictionPolicy` defines the algorithm to be used in case an element needs to be evicted from the cache. Possible values are:

- LRU - least recently used
- LFU - least frequently used
- FIFO - first in first out

Option: statistics

The option `statistics` defines whether the cache should capture statistical information, and if yes at which accuracy level. The statistics comprise information like cache hits, cache misses and average time to get an element. Possible values are:

- OFF - no statistics
- ACCURACY_NONE - fast but not accurate
- ACCURACY_BEST_EFFORT - best effort accuracy
- ACCURACY_GUARANTEED - guaranteed accuracy

Configuring the Email Server

Certain facilities within CentraSite communicate information to users using email (the Send Email Notification policy action, for example). These facilities will not function properly until you configure CentraSite's email settings. These settings specify the Simple Mail Transport Protocol (SMTP) server that CentraSite is to use for sending outgoing email messages.

Configuring the Email Server Settings

Use the following procedure to specify the email server settings for CentraSite. To perform this procedure, you must know the name (or IP address) of the email server that CentraSite is to use and the port number on which that server listens for SMTP requests. If the email server is configured to authenticate users, you must additionally provide the user ID and password that CentraSite is to use to log on to the server.

You can configure the email server settings by executing the following commands in the command line interface `CentraSiteCommand.cmd` (Windows) or `CentraSiteCommand.sh` (UNIX) of Command Central. The command line tool is located in <*SuiteInstallDir*>/CentraSite/utilities.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

To configure the email server settings

1. Create an XML configuration file that contains the following predefined properties. This file should be in Java XML properties format. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
  <properties version="1.0">
     <entry key="com.centrasite.config.email.SMTPHost">
      localhost</entry>
     <entry key="com.centrasite.config.email.SMTPPort">
      25</entry>
     <entry key="com.centrasite.config.email.ReplyTo">
      noreply@editthisdomain.com</entry>
     <entry key="com.centrasite.config.email.ConnectionTimeout">20</entry>
     <entry key="com.centrasite.config.email.Authentication">false</entry>
     <entry key="com.centrasite.config.email.User">xyz</entry>
     <entry key="com.centrasite.config.email.Password">xyz</entry>
     <entry key="com.centrasite.config.email.TransportLayerSecurity">
      false</entry>
  </properties>
```

   Descriptions of these properties are as follows:

In this field...	Specify...
SMTPHost	The name or IP address of the machine on which the SMTP server is running.
SMTPPort	The port on which the machine specified in **SMTP Host** listens for SMTP requests.
ReplyTo	The email address to which responses to the emails sent by CentraSite are to be directed. CentraSite uses this address to populate the "From" email header in the emails that it sends.
ConnectionTimeout	The number of seconds that CentraSite will wait for the email server to accept a connection request.

In this field...	Specify...
	If the email server does not respond within the specified time period, CentraSite writes an error message to the console and discards the email.
Authentication	Whether the SMTP server authenticates users. Set **Authentication** to `yes` if authentication is enabled on the SMTP server. If you set this field to `yes`, you must specify the appropriate log-on credentials in the **User** and **Password** fields below.
User	The user ID that CentraSite is to use to log on to the SMTP server. This value is required only when **Authentication** is set to `yes`.
Password	The password that CentraSite is to use when it logs on to the SMTP server. This value is required only when **Authentication** is set to `yes`.
TransportLayerSecurity	If true, enables the use of the `STARTTLS` command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. Note that an appropriate trust store must configured so that the client will trust the server's certificate. Defaults to false.

2. Execute the following command in this format:

```
CentraSiteCommand set Email [-url <CENTRASITE-URL>] -user <USER-ID>
-password <PASSWORD> -file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `set Email` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.

Parameter	Description
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

For example:

```
CentraSiteCommand set Email -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -file config.xml
```

Getting the Email Server Settings

You can get (retrieve) the email server settings by executing a command in the command line interface of Command Central.

To get the settings, execute the following command, which will print the configuration:

```
CentraSiteCommand.sh get Email [-url <CENTRASITE-URL>] -user <USER-ID>
-password <PASSWORD> -file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `get Email` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

For example:

```
CentraSiteCommand.sh get Email -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -file config.xml
```

Overview of the Administration Tools

Overview of the Command Line Tool "inoadmin"

The command line tool `inoadmin` offers functionality for performing low-level administrative operations on the CentraSite Registry Repository. The functionality provided by inoadmin includes:

- Starting and stopping the CentraSite Registry Repository.
- Maintaining the internal database that houses the CentraSite Registry Repository.
- Configuring port numbers used by the CentraSite Registry Repository.
- Configuring Secure Communication between the CentraSite Registry Repository and the CentraSite Application Server Tier.

The tool is located in the directory <*SuiteInstallDir* >/CentraSite/bin. If you call inoadmin without parameters, you receive a summary of the command syntax and the available functions.

You can use environment variables to configure the inoadmin behavior:

Environment variable	Purpose
INOADMIN_OUTPUT_TXT	Determines whether the inoadmin output is displayed as an XML document or as a formatted table. If the value is null (i.e. if the value is not defined), the inoadmin output is displayed as an XML document. If any non-null value is specified, a formatted table is displayed.
INOADMIN_NO_ MESSAGE_OUTPUT	Determines whether progress status messages are output while inoadmin calls are running. If this variable is null (i.e. if the value is not defined), all progress status messages are output. If any other value is set for this variable, the progress status messages are suppressed and only the result is displayed.
INOADMIN_RUN_AS_JOB	Creates a job log that contains status information generated while inoadmin is running.

If you use inoadmin within a command script, you can use the return status to verify the successful completion of the command. A zero return status means that the command executed successfully, whereas a non-zero return status means that the command did not execute successfully.

Overview of Command Central

Software AG Command Central is a tool that release managers, infrastructure engineers, system administrators and operators can use to perform administrative tasks from a single location. Command Central can assist with the following configuration, management and monitoring tasks:

- Infrastructure engineers can see at a glance which products and fixes are installed where, and can easily compare installations to find discrepancies.
- System administrators can configure environments using a single web UI, command-line tool or API, so maintenance can be performed with a minimum of effort and risk.
- Release managers can prepare and deploy changes to multiple servers using command-line scripting for simpler, safer lifecycle management.
- Operators can monitor server status and health, as well as start and stop servers from a single location. They can also configure alerts to be sent to them in case of unplanned outages

For more information, see the Command Central documentation.

Overview of the Toolbox Script

The toolbox script assists in running the command line utilities in CentraSite's toolbox. This toolbox is a collection of useful jar files located in the directory <*SuiteInstallDir* >/CentraSite/bin. Due to possible classpath issues it might not always be possible to run these jar files on location via `java -jar`. If encountering such issues use the toolbox script `CentraSiteTollbox.[cmd|sh]` located in <*SuiteInstallDir* >/CentraSite/utilities. To invoke an existing toolbox script, use a command `CentraSiteToolbox` in the following format:

```
CentraSiteToolbox.[cmd|sh] <ToolboxSript> <ToolboxScript Parameter>
```

Example

```
CentraSiteToolbox.[cmd|sh] CentraSiteCachConfiguration.jar http://
localhost:53307/CentraSite/CentraSite Administrator manage DISPLAY
```

All necessary classpath settings are performed by the `CentraSiteToolbox` script.

Return Codes from Command Execution

The following table describes the return codes you might encounter when using the command line tool `inoadmin`.

Return Code	Description
0	Execution of the command was successful.
1	A required parameter was not specified.
2	The command includes an invalid parameter.
3	The command includes an invalid number of parameters.
4	The output that a command returned does not match the expected values specified with the `version` option.
5	The server was not registered.
6	The server already exists.
7	The state of the server is active.
8	The state of the server is inactive.
9	The server startup failed.
10	The specified file cannot be located. Make sure you have entered the correct path and file name.
11	A parameter specified an invalid name.
12	A parameter specified an invalid value.
13	An error occurred during command execution.
14	Access to the operating system file is denied.
15	The backup deletion failed.
16	The specified service cannot be found.

Return Code	Description
17	First time startup of the service failed.
18	First time startup of the service failed.
19	The directory name is not valid.
20	Indicates that the dbspace name is invalid.
21	An autorepair is pending.
22	An operating system file is locked by the server.
23	No space left on the database.
24	There is not enough memory to run the script.
25	The function called is not implemented.

2 Authentication Topics and LDAP

Overview of Authentication Topics and LDAP

Authentication is the process of validating that a user's login credentials (for example the user's certificate, or user ID and password) match the credentials known to the system. CentraSite can use a number of different data sources, known as domains, to validate a user's credentials; these currently include the following:

- An *internal* text file
- Microsoft Active Directory (AD), when used via LDAP
- LDAP

This document is intended for customers who wish to configure CentraSite's user authentication features.

Assumptions

If an external repository, for example LDAP, will be used, this topic assumes that it has already been set up and that you have the necessary expertise and privileges to perform administrative tasks. Usually, the use of CentraSite does not influence any design decisions that were made in setting up an external user repository; CentraSite just needs to know how to access the users and groups of the user repository.

Overview of User Repositories

A user repository is in general terms a set of user credentials (optionally including user certificates etc.), with the possible addition of information such as the groups to which a user belongs, the user's address, telephone number and the email address. Often, an enterprise implements a central user repository that can be used by applications throughout a network to authenticate users; when a user tries to log in to an application, the application issues a request to the user repository to check whether the user credentials that she supplied are valid. Usually the user repository is created and maintained separately from the applications that use it.

A newly-installed CentraSitesystem is configured to authenticate users against an internal text file. This is intended to enable an administrator to log in and modify the configuration as required to meet enterprise requirements; typically, and in particular if you are working in a distributed environment, where one or more Application Server Tiers and a separate Registry/Repository are involved, an external repository such as Active Directory or LDAP will form the core of the authentication process.

Selecting a User Repository for Authentication

Access to information stored in CentraSitegenerally requires a user name and password, to ensure that data can only be stored, modified or retrieved by authorized users. CentraSitesupports the following types of user repository:

- An internal text file
- LDAP (for example Sun, OpenLDAP, ADS)

CentraSitemaintains information about each kind of user repository in so-called *authentication configurations*. An authentication configuration specifies the type of user repository to be used and any parameters that are required to configure the user repository. CentraSiteis delivered with one predefined authentication configuration, namely the configuration to use an internal text file, and this configuration is the default configuration. You can define additional authentication configurations; also, you can set any one of the defined configurations to be the default configuration.

For information about defining authentication configurations and setting the default authentication configuration, see "Creating Authentication Configurations" on page 66.

In general, user authentication information is stored in the user repository, not in CentraSite. CentraSite can contain a copy of selected data fields from the user repository for each registered CentraSite user. The user information in the CentraSite user registry is stored in objects of the type `User`. You can associate a CentraSite user object with a user in a user repository. In this case you can map data fields from the user repository into the user object in the CentraSite registry. The data in the mapped data fields is visible when you display the user object in CentraSite.

Domain Names of User Repositories

Each user repository is uniquely identified by a domain name. A user in a user repository is uniquely identified by the combination of domain name and user name.

When you log in to CentraSite Control, you must supply the name of a domain in which you are registered and your user name, in the format *<DomainName>* *<UserName>*, for example, `Headquarters\JSmith`.

The domain name for an authentication configuration of type Internal is always `INTERNAL`. Since this name is fixed, there can be only one such configuration defined per instance of the CentraSiteregistry.

Default User Repository

While CentraSiteis running, there is always exactly one default user repository. When you install CentraSite, the default user repository is set to the internal text file. You can change the default to any other user repository for which an authentication configuration exists.

Users who are registered in the default user repository can omit the domain name when they log in. For example, if the domain `Headquarters` is the default domain and it contains a user whose user name is `JSmith`, then this user can log in as `JSmith` instead of `Headquarters\JSmith`. Users who are not registered in the default user repository must always use the format *<DomainName>* *<UserName>* to log in.

Creating Authentication Configurations

When CentraSite utilizes a user repository, certain connection parameters are required. The connection parameters are stored in an authentication configuration. If you need to work with more than one user repository (for example, a user repository for test purposes and a user repository for a production environment), you can define several authentication configurations.

At any given time, only one authentication configuration can be the default authentication configuration.

After creating or modifying the authentication settings, the new settings apply immediately to the CentraSite Registry/Repository.

Commands for Creating and Maintaining Authentication Configurations

Commands for creating and maintaining authentication configurations are available with the command line tool CentraSiteCommand. You can use this tool to perform the following tasks:

- Create an authentication configuration
- Modify an authentication configuration
- Delete an authentication configuration
- Set a default authentication configuration
- List the names of all defined authentication configurations
- List details of a specific authentication configurations
- Validate that an authentication configuration is correctly specified

For information on how to perform these tasks, see "Configuring the Authentication Settings" on page 36.

Specifying the Domain Name

Unless the type of authentication that you wish to specify is `INTERNAL`, the authentication configuration requires a domain name. This is the domain name that will be used to address the users who are authenticated against the specified user repository.

Important: When working with LDAP, the domain name should be the name of a specific domain controller (DC) node in the LDAP tree structure. There can be many DC nodes in an LDAP tree structure, and you must choose the DC node that is the deepest ancestor node (parent, grandparent etc.) of all of the user nodes. Here, "deepest" means furthest away from the LDAP tree's root node. For example, if the usernames in an LDAP tree structure are located in the LDAP path `uid=Username,ou=People,dc=mydomain,dc=com`, then both `dc=mydomain` and `dc=com` are ancestor DC nodes of the user nodes, but since `dc=mydomain` is deeper than `dc=com`, you should specify the domain name as `MYDOMAIN` and not `COM`. If the path to the user nodes does not include any DC nodes, specify the root node. For example, if a user's full path is `cn=Username,ou=People,ou=RnD,o=Company`, set the domain name to `Company`.

The domain name for an authentication configuration of type Internal is always `INTERNAL`. Since this name is fixed, there can be only one such configuration defined per instance of the CentraSite registry.

Mapping User and Group Fields

When you specify an authentication configuration, you specify the correlation between properties stored in the CentraSite JAXR-based model for the object type User and properties stored in the external user repository.

The JAXR-based properties stored in CentraSite for the object type User are organized according to the following structure:

```
description
organization
personName
 firstName
 middleName
 lastName
 fullName
postalAddresses
 postalAddress
     street
     streetNumber
     postalCode
     city
     stateOrProvince
     country
     postalScheme
emailAddresses
 emailAddress
     address
telephoneNumbers
 telephoneNumber
     countryCode
     areaCode
     number
     extension
     url
URL
```

The mappings are used in CentraSite Control when you create a new CentraSite user and wish to associate the user with a user in the external user repository and also when you click on **Synchronize** for a user in the CentraSite Control. The corresponding dialog in CentraSite Control for locating the external user definition includes a search capability in which you can specify the JAXR-based mapping properties mentioned above to locate a particular user. The search mechanism translates the JAXR-based property searches into corresponding searches of the properties of the external user repository, using the mappings you define here. For more information, see "Adding a User" on page 88.

Specify the mappings as required. Typically, you only specify mappings for properties that you wish to make available for searches of the external user repository. If you do not require the capability of searching the external user repository, you can leave all of the fields empty.

Note: User property mappings are not available for users who are stored in the internal repository.

Configuring the Internal Authentication Type

General

The Internal authentication type allows you to authenticate a user against a set of user names and passwords that are maintained in a text file on the CentraSiteRegistry/ Repository. Passwords are stored in SHA-512 hashed format; they cannot be decrypted. All user names and passwords are case-sensitive.

A typical use of such an authentication type would be during the initial set-up and testing of all required CentraSitecomponents. In a production environment, one would typically use a central repository, e.g. Microsoft Active Directory or LDAP, instead of Internal authentication.

The domain name for the Internal authentication type is always `INTERNAL`; this cannot be changed. A user who is registered in the text file can log in using the domain and name `INTERNAL\`*<UserName>*, where *<UserName>* is the registered user name.

The Internal user repository initially contains one predefined user named `Administrator` with the password `manage`. This user logs in using the domain and user name `INTERNAL\Administrator`. If your default authentication configuration is the Internal configuration, this user can log in using just the user name `Administrator`, without specifying the domain name explicitly.

Caution: As soon as possible after completing installation, you should change the password that is associated with the user `Administrator`!

The dialog for creating a configuration for Internal authentication asks for the following values:

Parameter	Description
`Domain ID`	The domain ID is always `INTERNAL`. This cannot be changed.
`Expiration`	The number of seconds that the user is cached in the server after successful authentication. Changes made to the user, e.g.

Parameter	Description
	deletion or password changes, do not take effect until this time has elapsed.

Administration of Users and Passwords for Internal Authentication

CentraSite provides a command line tool `internaluserrepo` to allow you to perform administration tasks on the internal authentication file, such as adding users, deleting users and changing passwords.

The command line tool is located at:

<*SuiteInstallDir*>\common\bin\internaluserrepo.bat (Microsoft Windows)

<*SuiteInstallDir*>/common/bin/internaluserrepo.sh (UNIX)

The text file is located at <*SuiteInstallDir*>/common/conf/users.txt on all systems.

The usage of the command line tool is as follows:

On Windows

```
internaluserrepo.bat [-f <filename>] [-c] [-p <password>] [-d | -e] <userId
```

On UNIX

```
./internaluserrepo.sh [-f <filename>] [-c] [-p <password>] [-d | -e] <userId
```

For more information about the usage of `internaluserrepo` command tool, see http://documentation.softwareag.com/webmethods/wmsuites/wmsuite9-7/Security_Infrastructure/9-7_Security_Infrastructure/using/Creating_Internal_User.htm.

Configuring Active Directory Server

Note: Using an Active Directory Server as the user repository is only supported via LDAP.

The dialog for Active Directory Server asks for the following values:

Parameter	Description
Domain ID	See "Specifying the Domain Name" on page 66.
Host	The name of the machine on which the Active Directory server is running.
Port	The port on which the Active Directory server is running. Only needed if it is not the default value, which is 389.

Parameter	Description
Forest DN	The base bind distinguished name for the node under which all the domains reside. Example: if there are domains like: dc=HR,dc=abc,dc=com and dc=USA,dc=abc,dc=com and dc=EUR,dc=abc,dc=com, then this parameter would be `dc=abc,dc=com`.
Expiration	The number of seconds that the user is cached in the server after successful authentication. Changes made to the user, e.g. deletion or password changes, do not take effect until this time has elapsed.

For more information, see "Creating Authentication Configurations" on page 66.

Configuring LDAP

Principles of Configuring Against LDAP

CentraSite supports various LDAP configurations and provides standard settings that allow you to set up your authentication quickly against these standard systems.

There are many questions that are involved when you configure against an LDAP system:

- What kind of LDAP server is it?
- What is the hierarchical node structure of the LDAP server?
- In which kinds of objects are the user and group definitions contained?
- Which node properties contain the user names or group IDs?
- What other property mappings are required?

In general, before you begin to specify the configuration, we recommend you to study the LDAP structure and contents using an LDAP browser. There are various freeware tools such as JXplorer that allow you to do this. Using the LDAP browser, you can bind to an LDAP server, then navigate through the hierarchy to see the structures that contains the users and groups. Also, you can open the nodes that contain the definitions of individual users or groups, and view the properties that are stored for each user or group. An example of a node for a user `testuser01` might show the following properties:

Property name	Value
cn	testuser01

Property name	Value
objectClass	OpenLDAPperson
Mail	JohnSmith@MyCompany.com
Phone	+1 234 555 678

The path to the node for this user might be com/People/Location3/testuser01, where com is the root node. The setup on this LDAP server might be that all users are stored under the People node (com/People/…) and all groups are stored under the Groups node (com/Groups/…). Since every CentraSite customer can define their LDAP user and group structures differently, the details of the LDAP configuration that you will perform in CentraSite vary accordingly, since you must map explicitly to the customer LDAP structures.

Performing the LDAP Configuration

The general values that you can specify for an LDAP configuration are described in the following table. For more information, see "Creating Authentication Configurations" on page 66.

Value	Description
Domain ID	See "Specifying the Domain Name" on page 66.
LDAP server (host:port)	This is the host name (server and domain) of the machine where the LDAP server is located. You can specify a *Host:Port* combination in this field, where *Port* is the port number of the LDAP server on the host machine. You can specify multiple hosts in this field, using the blank character as a separator, for example `Host:Port Host:Port ...` If you specify multiple hosts, they are tried in the given order until a connection can be established. Each host can also be specified with a scheme such as `ldap` or `ldaps`, using the syntax `ldap://Host:Port` or `ldaps://Host:Port`.
Server Type	This field allows you to specify the type of LDAP server that will be used.

Value	Description
	You can specify Active Directory as the server type if the Active Directory server is accessed via LDAP (e.g. from a UNIX system).
Caching time for user credentials	The number of seconds that the user is cached in the server after successful authentication. Changes made to the user, e.g. deletion or password changes, do not take effect until this time has elapsed. This setting is provided for performance reasons. The default value is 120 seconds. If the connection to the LDAP server is slow, you can increase this figure.

The user-specific settings that you can specify are the standard LDAP settings. Refer to the documentation of your LDAP system supplier for details. Here are some examples.

User-specific Value	Description	Example
DN	The directory tree part of the distinguished name (standard LDAP terminology) of the entry. The method of specifying the path uses the standard LDAP path convention: first, a unique property of the DN node is specified, along with the property's value. Usually the property `ou` (organizational unit) is the property chosen for this purpose. Then the next higher `dc` node (i.e. a node with a `dc` property), then the next higher `dc` node and so on, until finally the root node.	ou=people, dc=MyServer,dc=com This example identifies the node whose `ou` property has the value `people` and is located under the node whose `dc` property is `MyServer`, which in turn is located under the node whose `dc` property has the value `com`.
Object	This identifies a property value that is used to categorize nodes as user nodes. For example, if you specify `OpenLDAPperson`, this means that user nodes can be recognized by being of object class `OpenLDAPperson`.	inetOrgPerson
Group Attribute	If the user repository specifies a property linking users to	memberOf

User-specific Value	Description	Example
	the groups to which they belong, specify the name of the property here. If there is no such property, leave this field blank.	
Field	This is the name of the property in the user node that uniquely identifies the user. (The attribute name of he RDN of users.)	cn

The group-specific settings that you can specify are the standard LDAP settings. Refer to the documentation of your LDAP system supplier for details. Here are some examples.

Group-specific Value	Description	Example
DN	This is similar to the DN property for users, but identifies a DN node for groups rather than for users.	ou=Groups,dc=abc,dc=de
Object	This identifies a property value that is used to categorize nodes as group nodes. For example, if you specify `groupOfNames`, this means that group nodes can be recognized by being of object class `groupOfNames`.	groupOfUniqueNames
User Attribute	If the user repository specifies a property linking a group to the users who are members of the group, specify the name of the property here. If there is no such property, leave this field blank.	member
Resolution	This specifies whether group nodes contain links to the users who are members of the group, or whether user nodes contain links to the groups they belong to. The option `recurse down` means that group nodes contain links to users. The option	Recurse Up

Group-specific Value	Description	Example
	`recurse up` means that user nodes contain links to groups.	

Note: If you are using LDAP, note that only the `recurse up` option is supported for group resolution.

Technical Principal for LDAP

Background

CentraSite can only find and authenticate a user name via the LDAP mechanism if either:

- the user name is located directly beneath the LDAP node that represents all users (specified via the User DN configuration value – for example, if user names are in the form `uid=Username,ou=People,dc=mydomain,dc=com` then the user name must be beneath the node `ou=People,dc=mydomain,dc=com`), or:
- the LDAP server allows "anonymous bind".

The technical principal is a user name or user account that preferably should not belong to a real user; in other words, the technical principal is normally the ID of a fictitious user. It is intended for organizations that store their user entries in branched LDAP directory structures, for example `uid=Username,loc=Germany,ou=People,dc=mydomain,dc=com` but do not allow anonymous bind. The technical principal must be defined in LDAP as having (at least) read access to all users and groups that are to be used by CentraSite.

When CentraSite is configured to use this feature, *all* LDAP accesses take place using the technical principal. For example, if a user with user name `user1` and password `pwd1` wants to log in to CentraSite Control, LDAP is accessed using the technical principal and the record for the user `user1` is checked.

Example of Configuring LDAP Authentication

You can set up LDAP Authentication using the command line tool, CentraSiteCommand, which is located in <*SuiteInstallDir*>/CentraSite/utilities.

The command to start the command line tool is as follows. The example assumes that there is a user `AdminUser` who has the CentraSite Administrator role, and this user has the password `AdminPass`.

```
CentraSiteCommand.cmd set Authentication -domain LDAPDomain
```

The sample interactive dialog is as follows. During each step of the command, the server prompts you to enter the basic details for LDAP authentication.

```
Executing the command : set Authentication
============================================================
Step 1: Basic LDAP Host Information
-----------------------------------------------------------
url - URL of Server (ldap(s)://host:port): ldap://MyServer01:10389
alias - Description of the Configuration [LDAPDomain]:
Do you want to use the LDAP Technical User (Y/N) [N]: Y
prin - Technical User: AdminUser
cred - Password of technical user: AdminPass
Repeat configuration step, Continue, or End? (R/C/E) [C]:
===========================================================
Step 2: Basic User Information
-----------------------------------------------------------
uidprop - User name attribute: cn
personobjclass - Found object is a person: inetOrgPerson
userrootdn -  Location to be searched for users: ou=people,ou=gdm,o=sag
dnprefix - Prefix to attach in front of the username [cn=]:
dnsuffix - Suffix to attach after the username [,ou=people,ou=gdm,o=sag]:
memberinfoingroups - Search users in a group (Y/N) [N]: Y
mattr - Member Search Operation: uniqueMember
Repeat configuration step, Continue, or End? (R/C/E) [C]:
===========================================================
Step 3: User Properties Mapping
-----------------------------------------------------------
no mappings defined
Do you want to keep this mapping? (Y/N): [N]:
Please provide your custom mapping (leave blank or enter "--" to unmap)
personName:firstName: givenName
personName:middleName:
telephoneNumbers:telephoneNumber:extension:
telephoneNumbers:telephoneNumber:areaCode:
URL:
telephoneNumbers:telephoneNumber:countryCode:
postalAddresses:postalAddress:postalScheme:
telephoneNumbers:telephoneNumber:url:
personName:fullName: displayName
emailAddresses:emailAddress:address: mail
personName:lastName: sn
postalAddresses:postalAddress:stateOrProvince:
description:
postalAddresses:postalAddress:streetNumber: postalAddress
telephoneNumbers:telephoneNumber:number: telephoneNumber
postalAddresses:postalAddress:country:
postalAddresses:postalAddress:postalCode: postalCode
postalAddresses:postalAddress:city:
organization:
Repeat configuration step, Continue, or End? (R/C/E) [C]:
===========================================================
Step 5: Basic Group Information
-----------------------------------------------------------
gidprop - Group attribute: cn
groupobjclass - Found object is a group: groupOfUniqueNames
grouprootdn - Location to be searched for groups: ou=groups,ou=gdm,o=sag
Repeat configuration step, Continue, or End? (R/C/E) [C]:
===========================================================
Step 6: Basic Group Information Mapping
-----------------------------------------------------------
Please provide your group information mapping
description: description
Repeat configuration step, Continue, or End? (R/C/E) [C]:
Successfully executed the command : set Authentication
```

Performing Maintenance on Authentication Configurations

Testing an Existing Authentication Configuration

You can test whether an authentication configuration contains the correct values for accessing the user repository. This feature is only available for LDAP authentication configurations.

To test an LDAP authentication configuration, use the command line tool CentraSiteCommand with the option `validate Authentication`. For details of the tool syntax, see "Configuring the Authentication Settings" on page 36.

During the validation, CentraSite attempts to access the user repository and returns status messages indicating the following:

- Whether the user repository is currently accessible.
- Whether the user with the given password exists in the user repository.
- Whether the mappings for users are correct.
- Whether the references between groups and users are correct.

Some of the possible reasons for errors are listed below.

- Incorrect host, port number or protocol is specified in the host definition.
- Incorrect DN specified in user information.
- Incorrect object class specified in user information.
- Incorrect user field in user information.
- Invalid user-ID/password combination.

Editing an Existing Authentication Configuration

To edit an existing authentication configuration, use the command line tool CentraSiteCommand with the option `set Authentication`. For information about the tool syntax, see "Modifying a Configuration" on page 40.

Deleting an Existing Authentication Configuration

If you do not require a particular authentication configuration any more, you can delete it from the list of available configurations.

You cannot remove the pre-installed domain `INTERNAL`.

If you remove a configuration that is the current default configuration, the configuration is removed and the default reverts to the INTERNAL configuration.

To delete an existing authentication configuration, use the command line tool CentraSiteCommand with the option `remove Authentication`. For information about the tool syntax, see "Removing a Configuration" on page 41.

Note: When you delete an authentication configuration, CentraSite does not delete the user objects that are associated with this configuration. Thus, these users will still be displayed in the list of users in CentraSite Control, even though the domain to which they belong is no longer accessible to CentraSite.

Setting a New Default Authentication Configuration

If you have defined more than one authentication configuration, you can change the current default configuration to one of the other configurations.

The user domain of the new default configuration must include at least one user who is defined in CentraSite with the CentraSite Administrator role, otherwise you will be prompted to enter a user who will be defined as administrator in that configuration.

To set a new default authentication configuration, use the command line tool CentraSiteCommand with the option `set DefaultDomain`. For information about the tool syntax, see "Setting the Default Configuration" on page 40.

If the user domain of the configuration that you wish to set to the default does not contain any user who is defined in CentraSite with the CentraSite Administrator role, a dialog will appear, asking you to provide the user name and password of a domain user who will be granted this role in CentraSite.

If the user already exists in CentraSite, but does not have the CentraSite Administrator role, the role will be granted to the user. If the user does not exist in CentraSite, a user with the given user name will be created in CentraSite and will be granted the CentraSite Administrator role.

The dialog also allows you to specify an organization for the user, in cases where the user did not already exist in CentraSite. The newly created CentraSite user will be assigned to this organization. If you do not specify an organization, the user is assigned to the default organization.

Users who are in the default domain can log in without having to specify the domain name, but they can specify the domain name if they wish. Users who are not in the current default domain always have to specify the domain name when logging in.

Note: Keep the following in mind:

1. If your default authentication configuration contains only one user who has the CentraSite Administrator role in CentraSite, it is not possible to delete this user from CentraSite, or to remove the CentraSite Administrator role from the user. This is because the default configuration must always contain at least one user who is defined in CentraSite with the CentraSite Administrator role.
2. If you try to log in to a CentraSite component (for example, CentraSite Control) by supplying a user name and password but no domain name, the authentication mechanism assumes that you belong to the domain of the default configuration and

will authenticate you against this domain. If you change the default configuration as described above and subsequently try to log in to a CentraSite component, you must supply your domain name in addition to your user name, so that the authentication mechanism knows which domain to use to check your credentials.

When you set a new default authentication configuration, you might want to change the association between CentraSite users (that is, CentraSite registry objects representing users) and users in the external user repository. For information on how to do this, and in particular if you want to do this for many users, see "Reassociating Users" on page 94.

Creating an Administration User from the Command Line

As stated above, the user domain of the default authentication configuration must contain at least one user who is defined in CentraSite with the CentraSite Administrator role. Under certain circumstances, it might happen that such users are no longer available in the user repository, for example:

- if the user repository is currently not available (e.g. LDAP Server currently unavailable);
- if all of the users who have the CentraSite Administrator role in CentraSite have been deleted from the user repository.

In such cases, there are no users any more who can log in to CentraSite as a user with the CentraSite Administrator role, so no CentraSite administration tasks can be performed.

To resolve this problem, a mechanism is available to create a user in the user repository, assigned to the CentraSite Administrator role in CentraSite.

You can create an administration user by executing the following command in the command line interface CentraSiteCommand.cmd (Windows) or CentraSiteCommand.sh (UNIX) of Command Central. The tool is located in <*SuiteInstallDir*>/CentraSite/utilities.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

The syntax of the command is as follows:

```
CentraSiteCommand add Admin [-url <CENTRASITE-URL>] -user <USER-ID>
-password <PASSWORD> -domain <DOMAIN> -domainUser <DOMAIN-USER-ID>
-domainPassword <DOMAIN-PASSWORD> -organization <ORGANIZATION>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `add Admin` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user name of a user in the user repository who has the required rights to create a new user in the user repository.
`-password`	The password of the user identified by the parameter -`user`.
`-domain`	The domain name of the user repository associated with the configuration.
`-domainUser`	The user name for a new user to be created in the user repository. A user with this name will be created in CentraSite and will have the role CentraSite Administrator.
`-domainPassword`	The user repository password for the user specified in `domainUser`.
`-organization`	The organization to which the newly created CentraSite user will belong.

Example

```
CentraSiteCommand add Admin -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -domain LDAPDomain -domainUser AdminUser
-domainPassword AdminPass -organization MyOrganization
```

Logging of Login Authentication Messages

Purpose of Log Files

If you have configured your authentication settings but still experience problems when trying to log in, you can use CentraSite's log files to analyze the problem. Some log file entries contain information about authentication problems in general, whereas other log file entries contain information about authentication problems related to individual CentraSitecomponents.

You can configure the authentication logging mechanisms by modifying parameters in the file jaas.config, which is located in *<SuiteInstallDir>*/profiles/CTP/configuration. The parameters in this file allow you to make the following changes:

- Switch authentication logging on or off for all CentraSite components.
- Specify the depth of logging required.

Activating the Authentication Logging

The logging of authentication messages is controlled by properties you can set in the CentraSitemodule in the file jaas.config.

The CentraSitemodule consists of one or more actions, and each action introduced by a specification such as:

```
com.softwareag.security.sin.is.ldap.lm.LDAPLoginModule ...
```

This is the top-level authentication component, and any logging properties that you specify here apply to the logging for all SIN authentication components that the login module applies to. Note that the logging properties need only be applied to the first occurring login module.

The properties that you can specify for the top-level SIN component are:

- **useLog.** Specify `true` to switch logging on, or `false` to switch logging off.
- **logLevel.** Specify the level of logging information required. Possible values are:
  - error - log only error messages
  - info - log error and information messages
  - debug - log all messages with additional debug information
- **logFile.** Specify the path and file name of the log file.

The properties that you can specify for the individual authentication components are:

- **nativeLogLevel.** Specify the level of logging information required. You can specify a number from 0 to 6, with 6 providing the most logging information and 0 the least.
- **nativeLogFile.** Specify the path and file name of the log file.

Here is an example showing logging switched on for SIN and SSX:

```
CentraSite {
  com.softwareag.security.sin.is.ldap.lm.LDAPLoginModule required
    useLog="true"
    logFile="/opt/softwareag/profiles/CTP/logs/sin-SAG-LDAP.log"
    logLevel="DEBUG"
    domain="SAG"
    alias="SAG"
    applyDomain="true"
    url="ldap://daeqarh01.eur.ad.sag:10389"
    prin="cn=LdapUser4CSAdmin,ou=people,ou=gdm,o=sag"
    cred="manage"
    usecaching="false"
    useaf="true"
    dnprefix="cn="
    dnsuffix=",ou=people,ou=gdm,o=sag"
    userrootdn="ou=people,ou=gdm,o=sag"
    uidprop="cn"
    personobjclass="inetOrgPerson"
    mattr="uniqueMember"
    memberinfoingroups="true"
    grouprootdn="ou=groups,ou=gdm,o=sag"
    gidprop="cn"
    groupobjclass="groupOfUniqueNames"
    creategroups="true"
    createGroupProperties="true"
    createUserProperties="true";
};
```

This configuration creates the log file: /opt/softwareag/profiles/CTP/logs/sin-SAG-LDAP.log

The log shows whether login attempts are successful or not, and indicates the user domain where CentraSiteattempted to find the login user credentials, for example:

```
...Authenticator (<domain>, ...) was created successfully
...login of user <username> (domain: <domain>) was successful.
```

If the authentication was not successful, a message such as the following is displayed:

```
Login of user <username> (host: <hostname>, port:<portnumber>) failed.
```

Transforming and Migrating LDAP Configuration Data

If you are upgrading to CentraSite 9.7 from an earlier version of CentraSite, you must transform and migrate the LDAP configuration from the old Registry Repository to the new CentraSite JAAS configuration.

To resolve this problem, a mechanism is available to migrate the LDAP data in CentraSite.

You can transform and migrate the LDAP configuration to the new JAAS configuration by executing the following command in the command line interface CentraSiteCommand.cmd (Windows) or CentraSiteCommand.sh (UNIX) of Command Central. The tool is located in <*SuiteInstallDir*>/CentraSite/utilities.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

The syntax of the command is as follows:

```
CentraSiteCommand generate JaasConfiguration [-url <CENTRASITE-URL>]
-user <USER-ID> -password <PASSWORD>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `generate JaasConfiguration` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user name of a user in the user repository who has the required rights to create a new user in the user repository.
`-password`	The password of the user identified by the parameter –`user`.

Example

```
CentraSiteCommand add Admin -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage
```

The script generates JAAS LoginModule entries that correspond to the old LDAP configuration and saves the entries in the jaas.config file in the *<CentraSiteInstallDir>*/profiles/CTP/configuration directory. For each LDAP domain, the script creates user and group files that map internal (CentraSite) properties to external (LDAP) properties and saves the files in the *<CentraSiteInstallDir>*/profiles/CTP/configuration/com.softwareag.platform.config.propsloader directory.

Note: The `generate JaasConfiguration` utility transforms only domains of type LDAP. If you have advanced JAAS configurations such as single-sign-on configurations, you must migrate them manually.

Notes on Authentication in CentraSite

Case Sensitivity

User names and domain names are treated as either case sensitive or case insensitive, according to the configured authentication mechanism.

INTERNAL authentication	case sensitive
Active Directory authentication	case insensitive

LDAP authentication	case insensitive

Working in an Offline Environment

If you wish to work in an offline environment, for example on a laptop computer that is not connected to the network, you should be aware of certain restrictions that apply in the area of authentication.

Important: When CentraSite is installed in an environment where the users are authenticated against a central service, for example an LDAP server, authentication will not work if the machine is disconnected from the network. So if you intend to use CentraSite on a mobile device when it is not connected to the network, ensure that at least one user is available who can also be authenticated offline, for example from an internal or local LDAP user repository.

3 Users, Groups, Roles and Permissions

About Users

Users identify individuals that are known to CentraSite. You assign roles and permissions to users to specify which operations they can perform and which registry objects they can access.

When you initially install an instance of CentraSite, it has only two user accounts: an account for the *bootstrap user* and an account for the *default user*.

- *The bootstrap user* refers to the user who installs CentraSite. This user belongs to the Default Organization and becomes the initial Organization Administrator for the organization as well as its primary contact. This user is also given the CentraSite Administrator role, which gives him or her “super admin” privileges. After CentraSite is installed, you can assign other users to the Organization Administrator role and/or the primary contact position for the Default Organization.
- *The default user* represents an internal user that owns the pre-define objects installed with CentraSite. The default user exists for CentraSite’s internal use. You cannot edit or delete this account. You cannot use the default user account to log on to CentraSite.

Typically, the bootstrap user creates the initial set of organizations on the CentraSite registry/repository. Then, the organization administrators create user accounts for the users that belong to their organizations.

User Authentication Mechanisms

Although CentraSite maintains its own database of user accounts, the users associated with those accounts are authenticated by an external authentication system at log on time.

CentraSite is delivered with one predefined authentication configuration, namely the configuration to use an internal text file, and this configuration is the default configuration. However, after installation, you can configure it to also use the following types of authentication systems:

- The operating system's user repository
- Active Directory Server (ADS)

  Note: If the CentraSite registry/repository is installed on a UNIX or Linux machine, you can only use the Active Directory Server as the user repository if it is configured via the LDAP interface.
- Lightweight Directory Access Protocol (LDAP)

For more details, see "Overview of User Repositories" on page 64.

If you are working in a distributed environment, where one or more Application Server Tiers and a separate registry/repository are involved, you must configure CentraSite to use an external authentication system. If you are working in a mixed Windows and UNIX environment, CentraSite can use Active Directory or LDAP as the user repository for both.

Note: Although CentraSite allows you to define multiple user repositories for authentication, only one is the default at any given time. Users who log on to the system by just providing the user name will be authenticated against the default authentication system. If you wish to log on to CentraSite with a user name that does not reside in the default authentication system, you need to prefix the user name by the Domain ID that was defined for the respective authentication system.

Users defined in the external directory are not automatically entitled to log on to CentraSite. You must explicitly create users accounts for valid users on CentraSite as described in "Adding a User" on page 88.

For information about how to configure the authentication for CentraSite, see "Authentication Topics and LDAP" on page 63.

Note: Any change of the external user management is not synchronized with CentraSite. If a user is removed from the external user management (for instance on operating system level) the corresponding CentraSite user is not automatically deactivated. The CentraSite user associated with a deleted external user must be deactivated manually in CentraSite.

Active and Inactive Users

The users that you define in CentraSite can be *active* or *inactive*. An active user has an associated user account on the external authentication system and is permitted to log on to CentraSite. Inactive users exist in the registry, but they are not permitted to log on to CentraSite. Additionally, permissions cannot be granted to inactive users nor can ownership of assets be given to them (inactive users retain ownership of objects that already belong to them).

Administrators generally deactivate users that leave the company or otherwise cease to be valid users of the registry. Inactive users are also useful for representing individuals who figure prominently in your SOA environment, but are not direct users of CentraSite. For example, you might create users to represent individual members of a key steering committee. Although these individuals might never use CentraSite, by including them in the registry as users, you enable assets and other objects to be associated with those individuals. Furthermore, if the user definitions for these individuals include email addresses, an administrator can develop policies that send email alerts to these individuals when significant events occur in the registry. Points-of-contact for external entities such as suppliers and distributors are other individuals that you might want to model as inactive users.

Guest Users

CentraSite supports the concept of a *guest user*. Guests are users that can access the registry without a user account (i.e., they can log on to CentraSite as anonymous users). Generally, guest users are given read-only access to a controlled set of assets.

In CentraSite, the capabilities given to guests are determined by the set of permissions specified in the Guest role. By default, CentraSite allows a guest to use the Asset Catalog screens in CentraSite Control. When they use the Asset Catalog screens, guest can see any asset on which the system-defined group called "Everyone" has View permission. (In other words, when you want to give guest users the ability to see an asset, you grant View permission to the group Everyone.)

You can include additional permissions in the Guest role as is required by your site. You must do this with great care, of course. Any additional permissions you assign to this role will significantly increase the capabilities of an anonymous user.

For additional information about the Guest role, see "Predefined Roles in CentraSite " on page 132.

Who Can Create and Manage Users?

To create and manage (i.e., view, edit and delete) users for an organization, you must belong to a role that has the `Manage Users` permission. Users in the Organization Administrator role have this permission, although an administrator can assign this permission to other roles.

Note: Users that belong to a role that includes the Manage Organizations permission have the Manage User permission by implication. Such users can create and mange users in the organizations to which their Manage Organizations permission applies.

Adding a User

You can add users to CentraSite in any of the following ways:

- Using the **Add User** button on the Users page as described in "Adding an Individual User to CentraSite " on page 89. This procedure enables you to define a single user and associate that user with an account in the external authentication system.
- Using the **Bulk Load Users from External Source** button on the Users page as described in "Bulk Loading Users from the External Authentication System" on page 93. This procedure enables you to load multiple users from the external authentication system into CentraSite in a single step.
- Using the **Users** profile on the Edit Organization page as described in "Adding Users from the Organization's Users Tab" on page 94. This procedure enables you to load one or more users from the external authentication system into CentraSite in a single step.

Important: Do not begin adding users to CentraSite until after you configure CentraSite for the external authentication system that you intend it to use.

When you add a new user to CentraSite, keep the following points in mind:

- When you add a user from Active Directory or LDAP, CentraSite automatically populates many of the user's attributes in CentraSite with user information from the external authentication system. The exact set of attributes that CentraSite populates depends on how the user attributes in CentraSite have been mapped to the user properties in the external authentication system. For more information about mapping attributes, see "Authentication Topics and LDAP" on page 63.
- A user can belong to only one organization.
- Every user that you add to CentraSite automatically becomes a member of the Everyone group. This group represents the set of all users defined on CentraSite, including the guest user.
- If the user that you add has an associated user account on the external authentication system, CentraSite also does the following:
  - It adds the user to the following system-defined groups

Group	Description
Users	All users that belong to the user's organization.
Members	All users belonging to the user's organization or any of its descendants (i.e., children, children's children and so forth).

  - It assigns the Asset Consumer and Asset Provider roles to the user. (An administrator can modify these automatic role assignments, so the default role assignments for your organization might be different.)
  - It activates the user, i.e., it allows the user to log on to CentraSite components such as CentraSite Control, using the user ID and password stored in the external authentication system.

Adding an Individual User to CentraSite

Use the following procedure to add an individual user to an organization and optionally associate that user with an account in the external authentication system.

To add an individual user to CentraSite

1. In CentraSite Control, go to **Administration > Users > Users**.
2. Click **Add User**.
3. In the **Organization** field, specify the organization to which you want to add the user. (The drop-down list only displays organizations for which you have `Manage Users` permission.)

4. Click **Associate** to select the user that you want to add from the external authentication system. (Skip this step if the user you are adding to CentraSite represents an individual that will not log on to CentraSite).

   - If CentraSite is configured to authenticate users using the local OS user database and you need procedures for this step, see "Selecting Users or Groups from the Local OS User Database" on page 91.
   - If CentraSite is configured to authenticate users using Active Directory or LDAP and you need procedures for this step, see "Selecting Users or Groups from an Active Directory or LDAP Server" on page 92.

   Note that you can only search for users that are stored in the same repository as the user who is logged into CentraSite Control and is performing the current operation. For example, if your system has both internal users and LDAP users, an internal user cannot search for users that are stored in the LDAP repository.

5. Complete the following fields as necessary. (If you selected the user from an Active Directory or LDAP system, many of these fields will already be populated.)

In this field...	Do the following...
First Name	Specify the first name of the user.
Middle Name	*Optional*. Specify the middle name of the user.
Last Name	Specify the last name of the user.
E-mail Address	*Optional*. Specify the user's e-mail address. **Note:** Including an email address for a user makes it possible for CentraSite to notify the user of certain events using email.

6. On the **Address Information** tab, specify the following:

In this panel...	Do the following...
Address	*Optional*. Specify the user's address information.
Contact	*Optional*. Specify the phone and fax numbers for the user. You can specify multiple phone and fax numbers.

7. If you have any custom properties (key-value pairs) that you want to specify for the user, select the **Object-Specific Properties** profile and specify the key-value pairs as follows:

   a. Click the **Add Property** button.

   b. In the **Add Object-Specific Properties** dialog box, enter the name of the property and value for the property. You can add multiple values for a single property.

      - The name of the property can consist of letters, numbers and the underscore character (_). It cannot contain a space or other special characters.
      - You can optionally supply a namespace for the property.

   c. Click **OK**.

8. If an administrator has added custom attributes to the User type definition, select the **Attributes** profile and specify the attributes as necessary. Attributes that are marked with an asterisk (*) are required. You must at least specify all required attributes.

   Note: You will see the **Attributes** profile only if an administrator has added custom attributes to the User type definition.

9. Click **Save** to save the new user.
10. Update the **Groups** profile as necessary to add the user to additional groups. For procedures, see "Adding a User to a Group" on page 97.
11. Update the **Roles** profile as necessary to assign additional roles to the user. For procedures, see "Assigning Roles to a User" on page 98.

Selecting Users or Groups from the Local OS User Database

The following procedure describes how to use CentraSite's standard dialogs to search for users or groups in the local operating system's user database.

Keep the following points in mind when performing a search:

- This dialog opens to an empty list. You must type a search string to retrieve a list of users or groups. To retrieve all the users or groups, leave the **Search** field empty and click **Search.**
- Search strings are not case-sensitive .The search string “bar”, for example, will find “BAR” and “Bar”.
- Search strings are not accent-sensitive.
- When you are searching the user list, CentraSite searches the user ID attribute, not the user name attribute. Thus, if a user has the user ID `MyDomain\AdminUser01` and the name `John Smith`, a search for `Admin` will find the user, whereas a search for `John` will not.
- CentraSite does a “starts with” search on the user ID or group name. *The domain portion of the name is not included in the search*. Thus, if a user has the user ID `MyDomain\AdminUser01`, a search for `Ad` will find the user, whereas a search for `User01` or `My` will not.
- You can type the wildcard characters % or * alone in the Search field to retrieve the list of all users or groups. However, you cannot combine these wildcard characters

with other character sequences. The sequences “xx%” and “%xx”, for example, are not valid search strings.

- CentraSite automatically filters out users that have already been added to CentraSite. For example, if the local machine has users CH001 and CH002, and user CH001 has already been added to CentraSite, a search for users starting with CH, will return user CH002, but not user CH001.

To search the local OS user database

1. In the **Search** field, type a search string that specifies the characters with which the user ID begins. The following are examples.

If you type...	CentraSite will return...
`b`	User IDs that begin with b.
`bar`	User IDs that begin with bar.
`%`	All user IDs.
`*`	All user IDs.
`emptyString`	All user IDs.

2. Click **Search.**
3. Repeat steps 1 and 2 until you obtain a list that contains the users that you want to add to CentraSite.
4. Select the users or groups that you want to add to CentraSite.
5. If the user that you want to add to CentraSite is not known to the local system, but is known to a domain server to which the local operating system is connected, type the user’s domain-qualified name into the **Type Domain Name field**. (This field is not available in all versions of this dialog.)

   Note: If you type a user ID in the **Type Domain Name** field, CentraSite ignores any selections you have made in the user list.

6. Click **OK**.

Selecting Users or Groups from an Active Directory or LDAP Server

The following procedure describes how to use the standard dialogs to search for users or groups in an Active Directory or an LDAP server.

Keep the following points in mind when performing a search:

- CentraSite treats the text you enter as a partial string. For example, if you enter `al`, then `Alex`, `Allen` and `Salie` all fit the search criteria.

- You can use the asterisk (*) as a wildcard in the search text. CentraSite replaces the wildcard symbol with as many characters as necessary.
- Searches are not case sensitive.
- Searches are not accent-sensitive.
- The ADS or LDAP authentication system performs a user search based on the attribute mapping specified in the authentication configuration, and displays the users that fit the search criteria. For more information about authentication configurations, see "Creating Authentication Configurations" on page 66.

To search an Active Directory or LDAP server

1. In the **Search Criteria** panel, create the search criteria by selecting the attribute and the condition from the respective list boxes and typing the search string in the text box.
2. Select a search operator: "Equals" and "NotEquals". The "Equals" tests for attributes that are equal to a certain value. The "NotEquals" finds for attributes that do not have the same or equal value.
3. For advanced search using multiple attribute conditions, click the plus button and add a new condition for the search.
4. Specify the way in which the criteria are to be combined:
   - To specify that a user or group must meet all criteria to be considered a match, select **AND Condition**.
   - To specify that a user or group must meet at least one of the criteria to be considered a match, select **OR Condition**.
5. Click **Search**.
6. Select the users or groups you would like to add to the organization.
7. Click **OK**.

Bulk Loading Users from the External Authentication System

You use the following procedure to add multiple users from the external authentication system to CentraSite in a single step. You can specify which organization you want to add the users to.

To bulk load users into CentraSite

1. In CentraSite Control, go to **Administration > Users > Users**.
2. Click **Bulk Load Users from External Source**.
3. In the **Bulk Load Users from External Source** dialog box, select the users that you want to add to CentraSite.
   - If CentraSite is configured to authenticate users using the local OS user database, see "Selecting Users or Groups from the Local OS User Database" on page 91.

- If CentraSite is configured to authenticate users using Active Directory or LDAP, see "Selecting Users or Groups from an Active Directory or LDAP Server" on page 92.

4. In the field **Import to Organization**, specify the organization into which the users will be added.
5. Scroll through the user list to confirm that the selected users were added successfully.
6. Examine each new user that you added to the specified organization and update the user's attributes as necessary. (If you selected users from an Active Directory or LDAP system, many of the new users' attributes will already be populated.)

Adding Users from the Organization's Users Tab

You use the following procedure to add one or more users to CentraSite from your external authentication system.

To use this procedure, you must have `Manage Organizations` permission on the organization to which you want to add users.

To add users from an organization's Users tab

1. Open the Edit Organization page.
2. Select the **Users** profile and click **Add Users**.
3. In the **Add Users** dialog box, select the users that you want to add to CentraSite.
   - If CentraSite is configured to authenticate users using the local OS user database and you need procedures for this step, see "Selecting Users or Groups from the Local OS User Database" on page 91.
   - If CentraSite is configured to authenticate users using Active Directory or LDAP and you need procedures for this step, see "Selecting Users or Groups from an Active Directory or LDAP Server" on page 92.
4. Scroll through the user list to confirm that the selected users were added successfully.
5. Click **Save** to save the updated organization.
6. Examine each new user that you added to the organization and update the user's attributes as necessary. (If you selected users from an Active Directory or LDAP system, many of the new users' attributes will be populated already.)

Reassociating Users

Overview

If you have associated a CentraSite user with an external user, you may wish to change the association to a different external user.

This can be required, for example, if the responsibility for certain CentraSite assets moves from one person to another person in the same authentication domain. By reassociating the user, you can keep the name of the CentraSite user unchanged while changing to a new external owner.

Another possible use would be to handle user IDs when the default domain name changes, e.g. when switching from operating system authentication to LDAP authentication.

CentraSite provides a command line tool `ReassociateUsers` that allows you to reassociate one or more CentraSite users with new external user IDs. The script implemented as an executable jar and can only be run by a user who has the CentraSite Administrator role.

This command line tool reassociates CentraSite users with new external user IDs. Any permissions that were granted for the old external user ID will be modified to grant those permissions for the new external user ID.

Usage

Before you run the command line tool, create a database backup.

The tool consists of an executable jar file in the bin folder of the CentraSite installation. It requires a Java 6 runtime and needs to be called in the following way:

```
java -jar ReassociateUsers.jar <CentraSite DB URL> <administrator user id>
<password> <old user id> <new user id>
```

or

```
java -jar ReassociateUsers.jar <CentraSite DB URL> <administrator user id>
<password> <mapping file name>
```

For example:

```
java -jar ReassociateUsers.jar http://localhost:53307/CentraSite/CentraSite
DOMAIN\admin pAsSw0rD OLDDOMAIN\oldUser NEWDOMAIN\newUser
```

The first form (5 arguments) is for reassociating a single user, whereas the second form (4 arguments) is for reassociating multiple users in one execution of the tool.

When using the second form, the fourth argument specifies a text file that contains the user IDs. Each line of the mapping file contains one comma-separated pair of old and new user ID. A user ID must not occur more than once in these mappings.

The tool first checks for the following preconditions, which must all be met, otherwise the tool stops and no users will be reassociated:

- there is a unique registry object for the old user ID
- the old user ID can be uniquely identified in the security configuration
- there is no registry object for the new user ID
- there is no security configuration for the new user ID
- the reassociated user is a login user
- the reassociated user is not the CentraSite administrator user who is running the utility

- the domain of the new user ID must exist in the security configuration
- a GUI configuration does not exist for the new user ID

If all preconditions are met, the tool performs the reassociation. This process may take some time. The tool progress is reported to standard output.

Viewing the Users List

You use the Users page to view the list of users defined on CentraSite.

To view the users list

1. Do one of the following:
   - In CentraSite Control, go to **Administration > Users > Users** to view the list of all users that are defined in CentraSite.
   - Go to the Edit Organization page and choose the **Users** profile to view the list of users in that particular organization.
2. If you want to filter the list, type a partial string in the **Search** field. CentraSite applies the filter to the **Name** column.

If you type...	CentraSite Displays
`b`	Names that contain `b`
`bar`	Names that contain `bar`
`%`	All names

The users list provides the following information about each user:

Column	Description
Name	The name of the user.
User ID	The log in ID of the user.
Organization	The name of the organization to which the user belongs.
Can Log On	The status of the user.

Icon	Description
🟩	The user is active (can log on to CentraSite).

Column	Description	
	⊙	The user is inactive (cannot log on to CentraSite).

Viewing or Editing Information About a User

You use the Edit User page to examine or modify information about a user.

Note: Changing the value of the **Organization** field moves the user to the specified organization (without moving the user's assets). You can only change this field if you belong to the CentraSite Administrator role. For more information about how CentraSite processes the movement of a user to another organization, see "Moving a User to a Different Organization" on page 102.

How to View or Edit a User's Information

To view or edit a user's information

1. In CentraSite Control, go to **Administration > Users > Users**.
2. On the Users page, locate the user whose details you want to view or edit.
3. From the user's context menu, select the **Details** command.
4. View or edit the attributes on the Edit User page as necessary. For additional information about the attributes on this page, see "Adding an Individual User to CentraSite " on page 89.
5. If you have made any changes to the users, click **Save**.

How to View Details for Multiple Users

You can view details for multiple users as follows:

To view details for multiple users

1. In CentraSite Control, go to **Administration > Users > Users**.
2. Mark the checkboxes of the users whose details you want to view.
3. In the **Actions** menu, click **Details**.

   The **Details** view of each of the selected users is now displayed.

Adding a User to a Group

Use the following procedure to add a user to or remove a user from a locally managed group (i.e., a group whose membership is defined within CentraSite, not on the external authentication system).

To add a user to a group

1. Open the Edit User page for the user whose group assignments you want to edit.
2. On the Edit User page, choose the **Groups** profile and do the following:
   - To add a user to a group, click **Add User to Group** and select the groups to which you want to add the user.
   - To remove a user from a group, select the groups from which you want to remove the user and click **Remove**.

     Note: You cannot remove the user from any of the system-defined groups.
3. Click **Save**.

Assigning Roles to a User

Use the following procedure to assign a role to or remove a role from a user.

To assign roles to a user

1. Open the Edit User page for the user whose role assignments you want to edit.
2. In the Edit User page, choose the **Roles** profile and do the following:
   - To assign roles to the user, click **Assign Role** and select the roles that you want to give to the user.
   - To remove roles assigned to a user, select the roles that you want to remove and click **Remove**.
3. Click **Save**.

Viewing the Assets Available to a User

Use the following procedure to display the list of assets that a particular CentraSite user owns.

To view a user's assets from the Edit User page

1. Open the Edit User page for the user whose role assignments you want to edit.
2. In the Edit User page, choose the **Assets** profile, which displays the list of assets that the user currently owns.

Activating or Deactivating a User

CentraSite Control offers the ability to activate or deactivate a user.

Activating a user account changes its status to Activated and allows the user to log on to CentraSite Control. Deactivating a user account changes its status to Deactivated and denies the user the privilege to log on to CentraSite.

A deactivated user cannot be assigned permissions, execute policies or become owner of the new assets. Also, the deactivated user cannot be a part of the approval group. Furthermore, if a user who was part of an approval group or a user who is the only member of the approval group is deactivated, the policy with that particular approval group is itself marked as fail.

You usually deactivate a user to prevent that user from logging on to CentraSite (temporarily or permanently). You must also deactivate a user account in order to delete it.

When you activate or deactivate a user, keep the following points in mind:

- You cannot deactivate the only remaining user in the CentraSite Administrator role in the CentraSite registry/repository or the only remaining user in the Organization Administrator role within an organization.
- You cannot deactivate the user who is an authorized approver for an approval flow that is in the Pending state.

You can activate or deactivate users in any of the following ways:

- From the Users page.
- From the Edit User page.
- From the Edit Organization page.

How to Activate or Deactivate a User via the Users Page

To activate or deactivate a user via the Users page

1. In CentraSite Control, go to **Administration > Users > Users**.
2. On the Users page, enable the check box next to the name of the user that you want to activate or deactivate. (You can select multiple users.)
3. From the **Actions** menu, choose **Activate** or **Deactivate** as needed.
4. Verify that the user's state has changed by checking the icon in the **Can log on** column.

Icon	Description
🟩	The user is active (can log on to CentraSite Control).
🟥	The user is inactive (cannot log on to CentraSite Control).

How to Activate or Deactivate a User via the Edit User Page

To activate or deactivate a user via the Edit User page

1. Open the Edit User page for the user whom you want to activate or deactivate.
2. In the Edit User page, click the **Activate User** or **Deactivate User** button as needed.

How to Activate or Deactivate a User via the Edit Organization Page

To activate or deactivate a user via the Edit Organization page

1. Open the Edit Organization page for the organization to which the user belongs.
2. On the Users tab, enable the checkbox next to the name of the user that you want to activate or deactivate. (You can select multiple users.)
3. From the **Actions** menu, choose **Activate** or **Deactivate** as needed.

Deleting a User

Deleting a user permanently removes a user from the CentraSite registry/repository. When deleting a user, keep the following points in mind:

- You cannot delete an active user. You must deactivate the user before you delete it. For procedures, see "Activating or Deactivating a User" on page 98.
- You cannot delete a user if any of the following conditions exist:
  - The user functions as the primary contact of an organization.
  - The user owns one or more assets in CentraSite.
- Deleting a user from CentraSite does not delete the user from the external authentication system.
- Make sure that at least one active user with the CentraSite Administrator role *always* resides in the Default Organization. Even if you plan to switch CentraSite's user authentication from one domain to another (such as, from the OS to an Active Directory or LDAP domain), to prevent a system lockout, make sure you have at least one user in the CentraSite Administrator role defined on CentraSite.

How to Delete a User

To delete a user

1. In the CentraSite Control, go to **Administration > Users > Users** to display the users list.
2. Ensure that the user is inactive.
3. Enable the check box next to the name of the user that you want to delete.
4. Click **Delete**.

When you are prompted to confirm the delete operation, click **OK**.

User is permanently removed from the CentraSite registry/repository. If the user had an associated user account in the external authentication system, that account is not affected.

How to Delete Multiple Users in a Single Operation

You can delete multiple users in a single step. The rules described above for deleting a single user apply also when deleting multiple users.

Important: If you have selected several users where one or more of them are predefined users (such as bootstrap user, for example), you can use the **Delete** button to delete the users. However, as you are not allowed to delete predefined users, only users you have permission for will be deleted. The same applies to any other users for which you do not have the required permission.

To delete multiple users in a single operation

1. In CentraSite Control, go to **Administration > Users > Users** to display the policy list.
2. Ensure that the users are inactive.
3. Mark the checkboxes of the users that you want to delete.
4. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

   Each selected user is permanently removed from the CentraSite registry/repository. If the user had an associated user account in the external authentication system, that account is not affected.

Deleting a User from the Command Line

In some circumstances, a user object cannot be deleted because internal objects that reference it cannot be deleted. It can happen when there are internal references to a user object even though the user is no longer the owner of any assets. There can also be references to the user object in the audit log. In such circumstances, a user object can only be deleted by using a Java command line tool `CentraSiteDeleteUser` provided specifically for this purpose.

Important: This tool is for use by administrators only, and should be used if the standard deleting procedure is not successful. In particular, the tool does not activate any policies that you might have defined.

This command line tool deletes a user, after transferring ownership of all of the user's objects to another user (the "target" user). It also redirects to the target user all associations that referred to the user to be deleted. Any ACLs granting rights for the user to be deleted will be modified to grant those rights to the target user. If the user to be deleted was the primary contact of an organization, the target user will be assigned that role.

To make the ownership transfer visible in the audit logs, an `OWNERSHIPTRANSFERRED` event is created for every registry object that references the user object. The description of the auditable event includes the original owner and states that a delete user operation has been executed.

The tool's combined operation performs the following steps:

- Transfer ownership of objects to the target user
- Redirect internal references to the target user
- Transfer access rights to the target user
- Transfer group memberships to the target user
- Remove the GUI configuration
- Remove the user object

The tool consists of an executable jar file in the bin folder of the CentraSite installation. It requires a Java 6 (or above) runtime and needs to be called in the following way:

```
java -jar CentraSiteDeleteUser <CentraSite DB URL> <administrator user id>
<password> <id or key of user to be deleted> <id of target user>
```

Examples:

```
java -jar CentraSiteDeleteUser http://localhost:53307/CentraSite/CentraSite
DOMAIN\admin pAsSw0rD DOMAIN\oldUser DOMAIN\newUser
java -jar CentraSiteDeleteUser http://localhost:53307/CentraSite/CentraSite
DOMAIN\admin pAsSw0rD uddi:1e5aff10-f3e3-11df-86fc-a6e2fa0ea483 DOMAIN\newUser
```

Please note that the target user must be active before using this tool. The user to be deleted must be deactivated.

Restrictions

The operation to delete a user requires several steps that cannot run within a single transaction. This means every parallel running transaction will be able to see intermediate results. Therefore please ensure that no other activity is in progress while you run the tool. Moreover if there is a failure of any of the steps during the execution, the registry will have an intermediate state. The original state cannot be recovered by rolling back the complete operation.

Moving a User to a Different Organization

The organization to which a user belongs determines, among other things, the organization to which the user's assets are published (by default) and the organization whose asset catalog the user can view (by default).

If a user transfers to a department or work group in another organization within your enterprise, you use the **Move** command to mirror that change in CentraSite. When you move a user to another organization, you can also move the user's assets to the target organization or you can leave them with their current organization.

Who Can Move a User?

To move a user to another organization, you must belong to the CentraSite Administrator role.

What Happens When You Move a User to Another Organization?

When you move a user to another organization, CentraSite does the following:

- Records the user's organization change in the audit log.
- Removes the user from the system groups in the user's former organization and adds the user to the system groups in the target organization.
- Triggers pre- and post-update policies on the User object as appropriate.
- Transfers the assets that the user owns to the target organization, if specified.
- Sends a notification to the inbox of the moved user when the move is complete.

Effect of Moving a User on the Groups to Which the User Belongs

When you move a user to another organization, the user is removed from the following system groups in his or her former organization and added to these groups in the target organization:

- Users
- Members

The user retains all other group memberships.

Effect of Moving a User on the User's Access to Assets

Members of the Users group for an organization have implicit View permission on the organization's assets. Because CentraSite transfers users from one Users group to another during a move, the moved users lose implicit access to the assets in their former organization (except for the assets that they own) and receive implicit access to the assets in the target organization. If users require continued access to the assets in their former organization, consider granting the Asset Consumer role (in the former organization) to them after the move.

Important: If there are any explicit instance-level or role-based permissions assigned to the Users and/or Members groups in their former organization, be aware that users will also lose those permissions when they leave the organization.

Moving users to another organization does not affect any instance-level permissions or role-based permissions that are granted directly to their user accounts or to any non-system groups (i.e., groups besides Users and Members) to which they belong. Therefore, other than losing access to certain assets as a result of leaving the Users and/

or Members groups in their former organization, users continue to have access to the same set of assets as they had before the move.

Effect of Moving a User on the User's Assigned Roles

When you move users to another organization, they lose the roles that were assigned to the Users and/or Members groups in their former organization and gain the roles that are assigned to the Users and/or Members groups in the target organization. Other than this change, users retain all of their other role assignments.

Transferring Inactive Users

You can transfer active or inactive users.

Users That You Cannot Move

You cannot move the default user or any other internal user that is installed by CentraSite.

Moving the User's Assets to the New Organization

When you move a user to another organization, you can optionally move all of the user's assets to the target organization at the same time. If you choose to do this, CentraSite will process the transfer of those assets.

Note: Transferring a user and the user's assets is an "all or nothing" operation. If the transfer of any one asset fails, neither the user nor the user's assets are moved.

Policies That are Triggered When You Move a User

CentraSite treats the move operation as an update to the User object. Thus, moving a user to a different organization triggers the execution of pre-update and/or post-update policies that apply to User objects. If a pre-update policy fails, the user is not moved into the target organization.

Note: It is the policies of the *target organization* that CentraSite applies to the move.

How to Move a User to Another Organization

Moving an Individual User

Use the following procedure to move an individual user to a specified organization.

To move an individual user

1. In CentraSite Control, go to **Administration > Users > Users**.
2. Locate the user that you want to move, and from its context menu, select **Move**.
3. In the **Move User(s)** dialog box, select the organization to which you want to move the user.

If you want to filter the organization list, type a partial string in the search field.

4. If you want CentraSite to also transfer the assets owned by the selected user, enable the **Move Assets owned by the selected user(s) to the new organization** option.

   If you do not enable this option, the user's existing assets will remain in the organization to which they are currently assigned (the transferred user will continue to serve as their owner).

5. Click **OK**.

   Note: You can also move a user from the **Users** tab on the Edit Organization page and from the **Organization** field on the Edit User page. (Be aware that if you move a user using the **Organization** field on the Edit User page, you cannot move the user's assets at the same time.)

Moving Multiple Users (Bulk Transfer)

Use the following procedure to move multiple users to a specified organization.

Important: If you have selected several users where one or more of them are predefined users (such as bootstrap user, for example), you can use the **Move** button to transfer the ownership of all of the selected users. However, as you are not allowed to transfer ownership of predefined users, only users you have permission for will be transferred.

To move multiple users

1. In CentraSite Control, go to **Administration > Users > Users**.
2. Select the users that you want to move to a particular organization.
3. Click the **Actions** link and select **Move**.
4. In the **Move User(s)** dialog box, select the organization to which you want to move the selected users.

   If you want to filter the organization list, type a partial string in the search field.

5. If you want CentraSite to also transfer the assets owned by the selected users, enable the **Move Assets owned by the selected user(s) to the new organization** option.

   If you do not enable this option, the assets will remain in the organizations to which they are currently assigned (the transferred users will continue to function as owners of the assets even though the assets are not transferred to the target organization).

6. Click **OK**.

   Note: You can also move multiple users using the **Actions** link on the **Users** tab on the Edit Organization page.

About Groups

A *Group* describes a set of CentraSite users. The group always belongs to exactly one organization, but can contain users from different organizations. Groups are visible to all users.

A group can either be managed locally within CentraSite or can be imported from the external authentication system.

Groups can be used for many purposes within CentraSite, including:

- Granting roles to specified groups of users.
- Granting instance-based permissions to specified groups of users.
- Identifying the set of users who can approve certain types of requests.
- Identifying the users to which a particular policy action is to be applied.

CentraSite has three main types of groups:

- *System groups* are shipped with CentraSite. When a user is added to CentraSite, CentraSite automatically adds the user to a specified system group depending on the organization to which the user belongs. And when the user is deleted, CentraSite automatically removes that user from the group. The membership of these groups cannot be manually updated or deleted by an administrator. For more information, see "System Groups" on page 106.
- *Locally managed custom groups* are user-defined groups that are defined and maintained within CentraSite.
- *Externally managed custom groups* are user-defined groups that are imported from the external authentication system.

System Groups

The membership of the following system groups is managed automatically by CentraSite. When you add a new user to CentraSite, CentraSite automatically adds the user to these system groups. When you delete a user from CentraSite, CentraSite automatically deletes the user from these groups. You cannot delete or edit the membership of these groups yourself. You can, however, assign roles and instance-level permissions to these groups.

This system group...	Contains...
Everyone	All users.
Users	All users in an organization. Each organization in the registry/repository has a Users group. By default, the

This system group...	Contains...
	Asset Provider and Asset Consumer roles are assigned to this group, which gives these roles to every user in the organization.
Members	All users in an organization or any of its descendant organizations (children, children's children and so forth) Each organization in the registry/repository has a Members group.

Custom Groups

Custom groups are groups that you define in CentraSite. A custom group can contain users from any organization in the registry/repository.

You can create a custom group of any one of the following types in CentraSite:

- **An externally managed custom group.** This is a group that is imported from the external authentication system. You cannot manually change the membership of such a group within CentraSite; CentraSite maintains the membership of an externally managed custom group by automatically synchronizing with the external authentication system.

  After an externally managed group is created, you cannot switch the group to a "locally managed" type of group, nor can you associate it with a different group on the external authentication system.

  If the external group includes members who are not existing users of CentraSite, those members will not become CentraSite users as a result of adding the group to CentraSite. If you subsequently add those individuals to CentraSite, however, they will automatically become members of this group.

- **A locally managed custom group.** This is a group that consists entirely of users who are registered in CentraSite. The users must be active users. For details, see "Active and Inactive Users" on page 87.

  The membership of the group is maintained in CentraSite. You can perform administrative tasks manually on the group in CentraSite, such as adding or removing users from the group.

  If you have a locally managed group, you can switch it to an externally managed group. For details, see "Adding an Externally Managed Custom Group to CentraSite " on page 110.

CentraSite supports static groups and nested groups.

Note: If you are using LDAP, note that only the "recurse up" option is supported for group resolution. The "recurse down" option is not supported.

External Group Synchronization

When you import a group from CentraSite's external authentication system, CentraSite fetches the group's details from the authentication system and automatically synchronizes (updates) the group's membership on CentraSite.

Group synchronization occurs in the following cases:

- **When you initially import a group from the external authentication system.** This creates an externally managed custom group in CentraSite. When such an externally managed custom group is created, CentraSite queries the external system to determine which members of the group are registered users in CentraSite. Those users become members of the externally managed custom group on CentraSite.
- **When you add a user to CentraSite from the external authentication system.** Whenever a new user is added from the external authentication system, CentraSite queries the external system to determine in which groups the user is a member. If any of those groups have been imported into CentraSite, the user is automatically added to the corresponding groups in CentraSite.
- **When a user is deleted from a group in the external authentication system.** The removal of a user from a group can be done only in the external authentication system, and the change will be reflected in CentraSite when the synchronization occurs.

Example

Assume that the users User1, User2, User3, User4 and User5 are defined on the external authentication system, and do not belong to any group on the external authentication system. Assume that all of these users except User1 have already been imported from the external authentication system to CentraSite, but do not yet belong to any group in CentraSite. Now assume that a group called GroupA is created in the external authentication system, and GroupA has members User1, User2 and User3.

If GroupA is imported to CentraSite, the registered CentraSite users User2 and User3 become members of GroupA in CentraSite, as the membership of the group is maintained in external authentication system (User 1 is not registered in CentraSite, therefore it is not available as a member in Group A). We cannot add more users manually to GroupA in CentraSite, since CentraSite just refers to the external authentication system for the membership details. However, if User4 and User5 are added to GroupA in the external authentication system, they also become members of the GroupA in CentraSite when the automatic synchronization occurs.

In this scenario, User1 is not yet a member of GroupA in CentraSite, since User1 is not a registered user in CentraSite. To add User1 to the group in CentraSite, you need to define User1 as a user in CentraSite and associate this user with GroupA in the external authentication system.

Who Can Create and Manage Groups?

To create and manage (i.e., view, edit and delete) groups for an organization, you must belong to a role that has the `Manage Users` permission for the organization. Users in the Organization Administrator role have this permission, although an administrator can assign this permission to other roles.

Note: Users that belong to a role that includes the Manage Organizations permission have the Manage User permission by implication. Such users can create and mange groups in any organization to which their Manage Organizations permission applies.

Creating Custom Groups

There are three ways in which you can create custom groups in CentraSite:

- Add a locally managed custom group to CentraSite
- Add an externally managed custom group to CentraSite
- Bulk load groups from the external authentication system

Adding a Locally Managed Custom Group to CentraSite

Use the following procedure to add a locally managed custom group to CentraSite.

To create a locally managed group

1. In CentraSite Control, go to **Administration > Users > Groups**.
2. Click **Add Group**.
3. In the **Group Information** panel, specify the following fields:

In this field...	Do the following...
Name	Enter a name for the new group. A group name can contain any character (including spaces). **Note:** The group name must be unique within an organization.
Description	*Optional*. Enter a short description for the new group. This description appears when a user displays the list of groups on the CentraSite Control.
Organization	Specify the organization to which this group belongs. (The drop-down list only displays organizations for which you have `Manage Users` permission.)

In this field...	Do the following...
	Important: Choose the organization carefully. You cannot change this assignment after the group is created.

4. To add users to the group, do the following:

   a. Click **Add User**.

   b. Select the users that you want to add to the group.

      If you want to filter the list, type a partial string in the **Search** field. CentraSite applies the filter to the **Name** column.

If you type...	CentraSite displays...
`b`	Names that contain `b`
`%`	All names

   c. Click **OK**.

5. Update the **Roles** profile as necessary to assign roles to this group. If you need procedures for this step, see "Assigning Roles to a Group" on page 114.

   Important: Verify that the **Organization** field specifies the correct organization for this group before you proceed to the next step.

6. Click **Save**.

Adding an Externally Managed Custom Group to CentraSite

Use the following procedure to add an externally managed custom group to CentraSite.

When performing this procedure, keep the following points in mind:

- You do not need to assign a name to the group. The group name will be imported from the external authentication system. (If you assign a name in the group's **Name** attribute, it will be overwritten.)
- Do not assign users to the group using the **Users** tab. The members of this group will be specified by the external authentication system. (If any users appear on the **Users** profile when you perform this procedure, those users will be removed from the group.)
- If you have a locally managed group that you would like to switch to an externally managed group, you can open the group and then execute the following procedure starting with step 4. Be aware, however, that the group's current name and membership will be replaced by the name and membership of the imported group.

To create an externally managed custom group

1. In CentraSite Control, go to **Administration > Users > Groups**.

   The Groups page displays the list of system and custom groups for which you have permission.

2. Click **Add Group**.
3. In the **Organization** field, specify the organization to which this group belongs. (The drop-down list only displays organizations for which you have `Manage Users` permission.)

   Important: Choose the organization carefully. You cannot change this assignment after the group is created.

4. Click **Associate**.
5. In the **Associate Group** dialog box, select the groups that you want to add to CentraSite.
   - If CentraSite is configured to authenticate users using the local OS user database, see "Selecting Users or Groups from the Local OS User Database" on page 91.
   - If CentraSite is configured to authenticate users using Active Directory or LDAP, see "Selecting Users or Groups from an Active Directory or LDAP Server" on page 92.

     Important: Choose the external group with care. You cannot change this association after the group is created.

6. In the **Description** field, specify a descriptive comment or remark (optional).
7. Update the **Roles** profile as necessary to assign roles to this group. If you need procedures for this step, see "Assigning Roles to a Group" on page 114.
8. Click **Save**.

Bulk Loading Groups from the External Authentication System

You use the following procedure to add groups through the bulk load option. By this procedure, you can add one or more group(s) in a single step to your organization or to another specified organization.

To create group(s) and save it to CentraSite

1. In CentraSite Control, go to **Administration > Users > Groups**.

   CentraSite displays the list of groups for which you have permission.

2. Click the **Bulk Load Groups from External Source** button.
3. In the **Bulk Load Groups from External Source** dialog box, select the groups that you want to add to CentraSite.

- If CentraSite is configured to authenticate users using the local OS user database, see "Selecting Users or Groups from the Local OS User Database" on page 91.
- If CentraSite is configured to authenticate users using Active Directory or LDAP, see "Selecting Users or Groups from an Active Directory or LDAP Server" on page 92.

4. In the field **Import to Organization**, specify the organization into which the groups will be added.
5. Scroll through the groups list to confirm that the groups you selected were added successfully.
6. Examine each new group and update its **Description** field and its **Roles** profile as necessary.

Viewing the Groups List

You use the Groups page to view the list of groups.

To view the groups list

1. In CentraSite Control, go to **Administration > Users > Groups** to view the list of all groups that exist in CentraSite.

   The Groups page provides the following information about each group:

Column	Description
Name	The name of the group.
Organization	The name of the organization to which the group belongs.
Description	A short description about the group.

Viewing or Editing the Attributes of a Group

You use the Edit Group page to examine and/or edit the attributes of a group. When editing a group, keep the following points in mind:

- You cannot modify the system-defined groups (i.e., Everyone, Users and Members).
- You cannot modify the name or membership of an externally managed group.

To view or edit the properties of a group

1. In CentraSite Control, go to **Administration > Users > Groups**.
2. Locate the group whose attributes you want to view or edit.

3. From the group's context menu, select the **Details** command.
4. Examine or modify the properties on the Edit Group page as required.

Field	Description
Name	The name of the group. A group name can contain any character (including spaces).
Description	Additional comments or descriptive information about the group.
Organization	*Read-only*. The organization to which this group belongs.
Associated with External Group	The group on the external authentication system with which this group is managed. If an external group has already been associated with this group, this field cannot be modified. If an external group has not been associated with the group, you can use the **Associate** button to associate an external group with it. *Doing this will switch the group from a locally managed group to an externally managed group*. The group's current name and member ship will be replaced by the name and membership information from the external group.
Users	The settings on this profile identify the users that are assigned to the group. To edit this list, see "Editing the Membership of a Group" on page 113.
Roles	The settings on this profile identify the roles that are assigned to the group. To edit this tab, see "Assigning Roles to a Group" on page 114.

5. If you have edited the settings on the Edit Group page, click **Save** to save the updated group.

Editing the Membership of a Group

Use the following procedure to modify the membership of a locally managed custom group.

Note: You cannot modify the membership of a system group or an externally managed group. System groups are automatically maintained by CentraSite. Externally managed groups are maintained by the administrators of the external authentication system.

To modify the membership of a group

1. Open the Edit Group page for the group whose membership you want to modify. If you need procedures for this step, see "Viewing or Editing the Attributes of a Group" on page 112.
2. On the Edit Group page, choose the **Users** profile and, do the following:
   a. To add users to the group, click **Add User** and select the users that you want to add to the custom group. If you need procedures for this step, see "Adding a Locally Managed Custom Group to CentraSite " on page 109.
   b. To remove users from the group, select the users that you want to remove and click **Remove**.
3. When you have finished your edits, click **Save** to save the updated group.

Assigning Roles to a Group

Assigning roles to a group confers the permissions associated with the role to each member of the group.

To assign roles to a group

1. Open the Edit Group page for the group whose role assignments you want to modify. If you need procedures for this step, see "Viewing or Editing the Attributes of a Group" on page 112.
2. On the Edit Group page, choose the **Roles** profile and do the following:
   a. To assign roles to the group, click **Assign Role** and select the roles that you want to give to the group.
   b. To remove roles from a group, select the roles that you want to remove and click **Remove**.
3. Click **Save** to save the updated group.

Deleting a Group

You use the Groups page to delete one or more custom groups. When deleting a group, keep the following points in mind:

- Deleting a group from CentraSite does not delete the associated group from the external authentication system.
- You cannot delete a system-defined group (not even if you belong to the CentraSite Administrator role).

How to Delete a Group

To delete a group

1. In the CentraSite Control, go to **Administration > Users > Groups** to display the groups list.
2. Enable the check box next to the name of the group that you want to delete.
3. Click **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

   The group is permanently removed from the CentraSite registry/repository. If the group was associated with a group definition in the external authentication system, the group in the external system is not affected.

How to Delete Multiple Groups in a Single Operation

You can delete multiple groups in a single step. The rules described above for deleting a single group apply also when deleting multiple groups.

Important: If you have selected several groups where one or more of them are system groups, you can use the **Delete** button to delete the groups. However, as you are not allowed to delete predefined groups, only groups you have permission for will be deleted. The same applies to any other groups for which you do not have the required permission.

To delete multiple groups in a single operation

1. In CentraSite Control, go to **Administration > Users > Groups** to display the groups list.
2. Mark the checkboxes of the groups that you want to delete.
3. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

   The selected group is permanently removed from the CentraSite registry. If the group was associated with a group definition in the external authentication system, the group in the external system is not affected.

About Roles and Permissions

In CentraSite, access control is enabled through a system of *roles* and *permissions*. A permission enables a user to perform a specified operation on a specified object. Permissions also enable users to work with specified parts of the CentraSite Control user interface. A role is a collection of permissions that can be assigned to an individual user or a group. The roles to which you belong and the permissions they includes dictate what portions of the CentraSite Control user interface you see, what objects you can work with, and what operations you can perform.

About Permissions

A permission enables a user to perform a specified operation on a specified object. Permissions also enable users to work with specified parts of the CentraSite Control user interface. It is largely CentraSite's system of permissions and roles that enables it to manage and maintain the separation of organizations in the registry.

Within CentraSite, there are two basic types of permissions: *instance-level permissions* and *role-based permissions.*

- Instance-level permissions enable access to one specific instance of an object. They provide very fine-grain control over objects in the registry. You can use an instance-level permission, for example, to give one specific user the ability to modify one particular asset in the catalog. *Instance-level permissions are granted directly to individual users or to groups.*
- Role-based permissions enable access to an entire class of objects or give users the ability to perform certain general operations in CentraSite. Role-based permissions provide coarse-grain control over the objects in the registry. You might use role-based permissions, for example, to allow a group of users to edit all of an organization's assets. Role-based permissions are not granted directly to individual users or groups. *Role based permissions are assigned to roles, and the roles are assigned to individual users or groups.*

Instance-Level Permissions

An instance-level permission enables access to:

- A specific instance of an object in the registry
- A specific folder in the repository
- A specific file in the repository

Instance-level permissions are granted at the following levels: View, Modify and Full.

The following table describes the capabilities that each level gives to a user. Note that each level of access inherits the capabilities of the preceding level. That is, each level implicitly includes the lower levels of access. In CentraSite's permission model, it is not possible to give a user delete permission without also giving that user modify permission. The Full permission, which enables a user to delete an object, implicitly gives the user Modify and View permissions as well.

Permission Levels on Registry Objects

This permission level...	Enables a specified user or group to do the following...
View	Read the object.

This permission level...	Enables a specified user or group to do the following...
Modify	Read and edit the object; view the object's instance-level permissions.
Full	Read, edit and delete the object; modify the object's instance-level permissions.

Permission Levels on Repository Folders

This permission level...	Enables a specified user or group to do the following...
View	Read and browse the folder and its properties.
Modify	Read and browse the folder and its properties; create files and subfolders in the folder; view the folder's instance-level permissions.
Full	Read and browse the folder and its properties; create files and subfolders in the folder; delete files and subfolders in the folder; modify the folder's instance-level permissions.

Permission Levels on Repository Files

This permission level...	Enables a specified user or group to do the following...
View	Read the file and its properties.
Modify	Read and edit the files and its properties; view the file's instance-level permissions.
Full	Read and edit the file and its properties; modify the file's instance-level permissions. **Important:** Full permission on a repository file does not give a user the ability to delete the file. Permission to delete a file is given only to those who have Full permission on the folder in which the file resides.

On Which Objects Can You Assign Instance-Level Permissions?

You can assign instance-level permissions to the following types of registry and repository objects:

- Assets (of any type)
- Design/change-time policies
- Run-time policies
- Report templates
- Taxonomies
- Repository folders
- Repository files

Implicit View Permissions on Registry Objects

CentraSite grants implicit View permission to all users on the following types of registry objects:

- Organizations
- Users
- Groups
- Roles
- Design/change-time policies
- Taxonomies

The View permission that CentraSite grants to users on these objects is permanent and irrevocable. It cannot be taken away from users through any instance-level or role-based permission.

Who Can Assign Instance-Level Permissions?

To assign instance level permissions to registry or repository objects, you must have Full permission on the object itself or belong to a role that includes a permissions that effectively grants you Full access to the object.

By default, a user has implicit (and irrevocable) Full permission on all of the objects that he or she owns. Consequently, the object owner is one user that is always permitted to set the instance-level permissions on an object.

How to Assign Instance-Level Permissions

For information about how to set instance-level permissions on registry and repository objects, see the following procedures:

For procedures on...	See...
Assets	*Working with the CentraSite Business UI* and *CentraSite User's Guide*

For procedures on...	See...
Report templates	*CentraSite User's Guide*
Design/change-time policies	*CentraSite User's Guide*
Run-time policies	*Run-Time Governance with CentraSite*
Taxonomies	"Setting Permissions on a Taxonomy" on page 334

Role-Based Permissions

Role-based permissions enable access to an entire set of objects or give users the ability to perform certain operations in CentraSite. Unlike instance-level permissions, which are assigned directly to individual users or groups, role-based permissions are assigned to roles, and roles are assigned to users and groups. You cannot assign a role-based permission to a user or group directly.

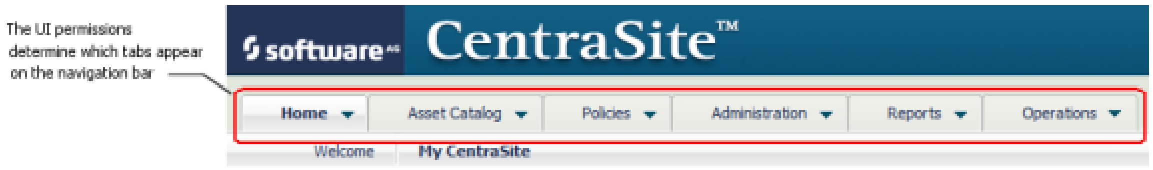
There are two basic types of role-based permissions:

- Permissions that enable access to areas of the CentraSite Control user interface.
- Permissions that enable users to create and/or work with certain types of registry and repository objects.

Some permissions are granted implicitly when you assign other permissions.

Permissions That Enable Access to Areas of the User Interface

Role-based permissions include a set of permissions that enable access to certain features and screen sets in the CentraSite Control user interface. These permissions determine which tabs a user sees in the navigation bar in CentraSite Control. For example, CentraSite will not display the navigation bar's **Policies** profile to a user unless the user has the Use the Policy UI permission. The UI-related permissions also include permissions that enable users to view certain logs and use certain controls in CentraSite Control.



Note: The user interface permissions apply only to the CentraSite Control user interface. They *do not* affect access to features or screen sets in the CentraSite plug-in for Eclipse.

By default, all CentraSite users have implicit (and irrevocable) permission to use the Asset Catalog area of the CentraSite Control user interface. To use the other parts of the user interface, a user must belong to a role that includes the appropriate user interface permission either explicitly (i.e., the role includes the permission itself) or implicitly (i.e., the role includes a permission that implicitly grants the UI permission). For more information about implicitly granted permissions, see "Implication of Additional Permissions" on page 123.

The following table describes the user interface permissions that are available in each CentraSite edition:

Available Permissions in CentraSite full-feature edition	Available Permissions in Community Edition
Use the Home UI	✓
Use the Policy UI	
Use the Administration UI	✓
Use the Reports UI	✓
Use the Operations UI	
View Policy Log	
View Approval History	
Register as Consumer	✓

Permissions That Enable Access to Objects

Role-based permissions include a second type of permissions that enable users to create and/or work with an entire class of objects.

Generally speaking, these types of role-based permissions grant a specified level of access on objects of a specific type. For example, the Modify Assets permission grants Modify-level access on all objects of the type Asset. Role-based permissions enable you to apply access controls over an entire class of objects instead of assigning permissions on each object individually.

Levels of Access Granted by the Role-Based Permissions

If a role-based permission grants access to an object, the name of the permission includes one of the following terms to indicate which level of access the permission provides.

If the name includes the following term...	The permission grants the following levels of access...
View	Read objects of a specified type. This level is equivalent to giving a user View instance-level permission on all objects of a given type.
Modify	Read and edit objects of a specified type. This level is equivalent to giving a user Modify instance-level permission on all objects of a given type.
Create	Create and read objects of a specified type. This level is equivalent to giving a user View instance-level permission of all objects of a given type and giving them the ability to create new instances of that type.
Manage	Create, read, edit, delete and modify the instance-level permission of objects of a specified type. This level is equivalent to giving a user Full instance-level permission of all objects of a given type.

Be aware that CentraSite does not provide role-based permissions at all levels for all object types. Access to certain objects types can only be granted at the Manage level, for example.

System-Level vs. Organization-Level Permissions

Role-based permissions that enable access to objects are either *organization-specific* or *system-wide*.

System-Level Permissions

A system-wide permission grants access to objects that are available to all organizations, such as taxonomies and asset types. Additionally, some system-wide permissions grant access to all objects of given type *in any organization in the registry/repository*.

The following table describes the system-level permissions that are available in each CentraSite edition:

Available Permissions in CentraSite full-feature edition	Available Permissions in Community Edition
Manage Organizations	✓
Manage System-wide Lifecycle Models	

Available Permissions in CentraSite full-feature edition	Available Permissions in Community Edition
Manage System-wide Design/ Change-Time Policies	
Manage System-wide Runtime Policies	
Manage Report Templates	✓
Manage System-wide Roles	✓
Manage UDDI Subscriptions	✓
Create UDDI Subscriptions	✓
View UDDI Subscriptions	✓
Manage Federations	
Manage Taxonomies	✓
Manage Asset Types	
Manage Runtime Targets	
Manage Runtime Event Types	
Manage Supporting Documents	✓
View Supporting Documents	✓

Organization-Level Permissions

An organization-specific permission grants a specific level of access to all objects of a given type *within a specified organization*. Permissions that enable access to assets, policies and life cycle models are organization-specific.

The following table describes the user interface permissions that are available in each CentraSite edition:

Available Permissions in CentraSite full-feature edition	Available Permissions in Community Edition
Manage Assets	✓
Create Assets	✓
Modify Assets	✓
View Assets	✓
Manage Design/Change-Time Policies	
Manage Run-Time Policies	
Manage Lifecycle Models	
Manage Users	✓
Manage Organizations	✓

Implication of Additional Permissions

Most role-based permissions grant additional permissions by implication. That is, the permission grants not only the permission that its name indicates, it grants additional implied permissions.

For example, any permission that grant access to a particular type of object automatically includes (by implication) the UI permissions required to work with that object. Users that belong to roles that include the Manage Taxonomy permission, for instance, automatically receive the Use the Administration UI permission by implication.

The Manage Organizations permission is an example of another permission that grants additional permissions by implication. This permission, when given at the system-level, essentially implies all other role-based permissions. Consequently, a user with Manage Organizations permission at the system-level can manage virtually any object in any organization.

Role-Based Permissions in CentraSite

The following table describes the individual permissions and their implied permissions in CentraSite, if any.

Permission	Scope	Purpose	Implied Permissions
Use the Home UI	System-wide	Grants the right to use the Home area in CentraSite Control. Without this permission the UI will hide the **Home** top-level navigation item.	None
Use the Policy UI	System-wide	Grants the right to use the Policy area in CentraSite Control. Without this permission the UI will hide the **Policy** top-level navigation item.	None
Use the Administration UI	System-wide	Grants the right to use the Administration area in CentraSite Control. Without this permission the UI will hide the **Administration** top-level navigation item.	None
Use the Reports UI	System-wide	Grants the right to use the Reports area in CentraSite Control. Without this permission the UI will hide the **Reports** top-level navigation item.	None
Use the Operations UI	System-wide	Grants the right to use the Operations area in CentraSite Control. Without this permission, the UI will hide the **Operations** top-level navigation item.	None

Permission	Scope	Purpose	Implied Permissions
View Policy Log	System-wide	Grants the right to view the policy log.	Use the Administration UI
View Approval History	System-wide	Grants the right to view the approval history.	Use the Administration UI
Register as Consumer	System-wide	Grants the right to register as a consumer of assets.	None
Manage Assets	Organization	Grants the right to manage assets and supporting documents within an organization.	Create Assets Modify Assets View Assets
Create Assets	Organization	Grants the right to create new assets within an organization.	None
Modify Assets	Organization	Grants the right to modify all assets and supporting documents within an organization. The `Modify Assets` permission is important not just for applying updates to existing assets but also for performing consumer registrations.	View Assets
View Assets	Organization	Grants the right to view all assets and supporting documents within an organization.	None
Manage Design/ Change-Time Policies	Organization	Grants the right to manage design/ change-time policies within an organization.	Use the Policy UI

Permission	Scope	Purpose	Implied Permissions
		Additionally, this implies the right to manage all policy-related objects such as policy conditions and policy parameters. Note that there is no explicit permission for viewing design/change-time policies, since design/change-time policies are visible for everyone.	
Manage Run-Time Policies	Organization	Grants the right to manage run-time policies within an organization. Additionally, this implies the right to manage all policy-related objects such as policy conditions and policy parameters. Note that there is no explicit permission for viewing run-time policies, since run-time policies are visible for everyone.	Use the Policy UI
Manage Lifecycle Models	Organization	Grants the right to manage lifecycle models (LCMs) within an organization. Note that there is no explicit permission for viewing LCMs, since LCMs are visible for everyone.	Use the Administration UI Modify Assets Manage Design/Change-Time Policies Manage Runtime Policies

Permission	Scope	Purpose	Implied Permissions
Manage Users	Organization	Grants the right to manage users of an organization. Additionally, this grants the right to manage groups and roles within an organization.	Use the Administration UI
Manage Organizations	Organization	Grants the right to manage an organization and all its child organizations. Also grants the right to manage the organization folder in Supporting Document Library (SDL). By default, the content of an organization folder is visible for every user of that organization. Note that all organizations are visible for everyone.	Manage Users Manage Design/ Change-Time Polices Manage Run-Time Policies Manage Lifecycle Models Manage Assets
Manage Organizations	System-wide	Grants the right to manage all organizations in the CentraSite instance. Note that all organizations are visible for everyone.	Manage Organizations (org-level permission for every organization) Manage System-wide Design/ Change-Time Policies Manage System-wide Run-Time Policies Manage-System-wide Lifecycle Models

Permission	Scope	Purpose	Implied Permissions
			Manage Report Templates
Manage System-wide Lifecycle Models	System-wide	Grants the right to manage lifecycle models. Note that there is no explicit permission for viewing system-wide lifecycle models, since system-wide lifecycle models are visible for everyone.	Use the Administration UI Manage Lifecycle Models (org-level permission for every organization) Manage System-wide Design/ Change-Time Policies Manage System-wide Runtime Policies
Manage System-wide Design/ Change-Time Policies	System-wide	Grants the right to manage design/ change-time policies. Additionally, this implies the right to manage the following policy-related objects: ▪ Action Categories ▪ Action Templates ▪ Action Parameters Note that there is no explicit permission for viewing system-wide design/change-time policies, since system-wide design/change-time policies are visible for everyone.	Use the Policy UI
Manage System-wide	System-wide	Grants the right to manage run-time policies.	Use the Policy UI

Permission	Scope	Purpose	Implied Permissions
Runtime Policies		Additionally, this implies the right to manage the following policy-related objects: ■ Action Categories ■ Action Templates ■ Action Parameters Note that there is no explicit permission for viewing system-wide run-time policies, since system-wide run-time policies are visible for everyone.	
Manage Report Templates	System-wide	Grants the right to manage system-wide report templates. Note that there is no explicit permission for viewing system-wide report templates, since system-wide report templates are visible for everyone.	Use the Reports UI
Manage System-wide Roles	System-wide	Grants the right to manage system-level roles. Note that there is no explicit permission for viewing system-wide roles, since system-wide roles are visible for everyone.	Use the Administration UI
Manage UDDI Subscriptions	System-wide	Grants the right to manage UDDI Subscriptions.	View UDDI Subscriptions Create UDDI Subscriptions

Permission	Scope	Purpose	Implied Permissions
Create UDDI Subscriptions	System-wide	Grants the right to create new UDDI Subscriptions and view existing UDDI Subscriptions.	View UDDI Subscriptions
View UDDI Subscriptions	System-wide	Grants the right to view UDDI Subscriptions.	Use the Administration UI
Manage Federations	System-wide	Grants the right to manage federations. Note that there is no explicit permission for viewing federations, since federations are visible for everyone.	Use the Administration UI
Manage Taxonomies	System-wide	Grants the right to manage taxonomies. Note that there is no explicit permission for viewing system-wide taxonomies, since system-wide taxonomies are visible for everyone.	Use the Administration UI
Manage Asset Types	System-wide	Grants the right to manage asset types. Note that there is no explicit permission for viewing asset types, since asset types are visible for everyone.	Use the Administration UI
Manage Runtime Targets	System-wide	Grants the right to manage run-time targets and target types. Note that there is no explicit permission	Use the Operations UI

Permission	Scope	Purpose	Implied Permissions
		for viewing run-time targets and target types, since run-time targets and target types are visible for everyone.	
Manage Runtime Event Types	System-wide	Grants the right to mange run-time event types. Note that there is no explicit permission for viewing run-time event types, since run-time event types are visible for everyone.	Use the Operations UI
Manage Supporting Documents	System-wide	Grants the right to manage the content of all folders in the Supporting Documents Library (SDL).	View Supporting Documents
View Supporting Documents	System-wide	Grants the right to view the content of all folders in the Supporting Documents Library (SDL).	None

Combining Role-Based and Instance-Level Permissions

When a user receives multiple permissions for the same object, the permissions are combined and the user receives the union of all the permissions.

For example, if you give a user instance-level View permission on an asset and that user belongs to a role that gives him or her Modify permission on the asset, that user will get View permission plus Modify permission on the asset (or, in effect, Modify permission since it implies View permission).

You will need to keep this concept in mind when granting role-based access to a large group of users (particularly to an entire organization). Anytime you use a role-based permission to give a group of users access to the entire set of assets in an organization, you can no longer use instance-level permissions to reduce the level of access for those users. For example, when everyone in your organization is given View Assets permission, you no longer have a way to use instance-level permission to selectively hide assets from certain users in the organization. In effect, the View permission becomes irrevocable for the users in the organization.

About Roles

A role is a collection of role-based permissions. Assigning a role to a user gives the user the permissions specified in the role. Roles can be assigned to individual users or to groups.

CentraSite provides many predefined roles for you to use. You can also create custom roles as needed.

Predefined Roles in CentraSite

CentraSite provides many predefined roles. These roles fall into two categories:

- *System-level roles*, which contain permissions for working with system-wide objects (i.e., objects that affect or are available to all organizations). Asset types and taxonomies are examples of system-wide objects.

  The following table identifies the predefined system-level roles that are available in each CentraSite edition:

Available Roles in CentraSite full-feature edition	Available Roles in Community Edition
CentraSite Administrator (CSA)	✓
Asset Type Administrator (ATA)	
Operations Administrator (OPA)	
Guest (G)	✓

- *Organization-level roles*, which contain permissions for working with organization-specific objects (i.e., objects that belong to and are used by one particular organization). Assets, policies and lifecycle models are examples of organization-specific objects. CentraSite maintains a set of organization-level roles for each organization in the registry/repository.

  The following table identifies the predefined organization-level roles that are available in each CentraSite edition:

Available Roles in CentraSite full-feature edition	Available Roles in Community Edition
Organization Administrator (OA)	✓

Available Roles in CentraSite full-feature edition	Available Roles in Community Edition
Asset Administrator (AA)	✓
Policy Administrator (PA)	
Asset Provider (AP)	✓
Asset Consumer (AC)	✓
API Runtime Provider	
API Publisher	
API-Portal Administrator	

The following table summarizes the permissions that CentraSite assigns to each of the predefined roles by default. Be aware that the permission assignments for most of these roles can be customized by an administrator. If an administrator has customized a predefined role at your site, it might have different permissions than what is shown here.

Permission	Type	Scope	System-Level Roles				Organization-Level Roles				
			CSA	ATA	OPA	Guest	OA	PA	AA	AP	AC
Use the Home UI	UI	System-wide	✓	✓	✓		✓	✓	✓	✓	✓
Use the Policy UI	UI	System-wide	✓		✓		✓	✓			
Use the Administration UI	UI	System-wide	✓	✓			✓				
Use the Reports UI	UI	System-wide	✓		✓		✓		✓	✓	✓

Permission	Type	Scope	System-Level Roles				Organization-Level Roles				
			CSA	ATA	OPA	Guest	OA	PA	AA	AP	AC
Use the Operations UI	UI	System-wide	✓		✓						
View Policy Log	UI	System-wide	✓	✓	✓		✓	✓	✓		
View Approval History	UI	System-wide	✓	✓	✓		✓	✓	✓		
Register as Consumer	UI	System-wide	✓				✓		✓	✓	✓
Manage Assets	ORG	Organization					✓		✓		
Create Assets	ORG	Organization					✓		✓	✓	
Modify Assets	ORG	Organization					✓		✓		
View Assets	ORG	Organization					✓	✓	✓	✓	✓
Manage Design/ Change-Time Policies	ORG	Organization					✓	✓			
Manage Run-Time Policies	ORG	Organization					✓	✓			

Permission	Type	Scope	System-Level Roles				Organization-Level Roles				
			CSA	ATA	OPA	Guest	OA	PA	AA	AP	AC
Manage Lifecycle Models	ORG	Organization					✓		✓		
Manage Users	ORG	Organization					✓				
Manage Organizations	ORG	Organization					✓				
Manage Organizations	SYS	System-wide	✓								
Manage System-wide Lifecycle Models	SYS	System-wide	✓	✓	✓						
Manage System-wide Design/ Change-Time Policies	SYS	System-wide	✓	✓	✓						
Manage System-wide Runtime Policies	SYS	System-wide	✓		✓						
Manage Report Templates	SYS	System-wide	✓		✓						
Manage System-wide Roles	SYS	System-wide	✓								

Permission	Type	Scope	System-Level Roles				Organization-Level Roles				
			CSA	ATA	OPA	Guest	OA	PA	AA	AP	AC
Manage UDDI Subscriptions	SYS	System-wide	✓		✓						
Create UDDI Subscriptions	SYS	System-wide	✓		✓						
View UDDI Subscriptions	SYS	System-wide	✓		✓						
Manage Federations	SYS	System-wide	✓								
Manage Taxonomies	SYS	System-wide	✓	✓							
Manage Asset Types	SYS	System-wide	✓	✓							
Manage Runtime Targets	SYS	System-wide	✓		✓						
Manage Runtime Event Types	SYS	System-wide	✓		✓						
Manage Supporting Documents	SYS	System-wide	✓								
View Supporting Documents	SYS	System-wide	✓								

System-Level Roles

The following predefined roles in CentraSite are system-level roles. Generally speaking, these roles contain permissions that enable users to work with system-wide objects. Some of these roles also enable users to manage (view, edit or delete) all instances of a given object type in any organization in the CentraSite registry/repository. For example, a user in the Centrasite Administrator role has, in effect, Full permission on every asset in every organization.

The main characteristic that distinguishes a system-level role from an organization-level role is that a system-level role is not generated for each new organization.

As noted in the following descriptions, some system-level roles can be edited and/or deleted and others cannot.

- **CentraSite Administrator.** The CentraSite Administrator role includes all available permissions. Users in this role have all the permissions to manage the complete system (that is, users with "superuser" permissions). The CentraSite Administrator only can change the permission set for a system role. Additionally, the CSA has permission to create and delete a system role. *This role cannot be modified or deleted*.
- **Asset Type Administrator.** Users in the Asset Type Administrator role can manage the type definitions and taxonomies in CentraSite.
- **Operations Administrator.** Users in the Operations Administrator role can manage the operational aspects of CentraSite. For example, the Runtime Event Types and Runtime Targets.
- **Guest.** Users in the Guest role can browse and use the asset catalog.

Organization-Level Roles

CentraSite automatically populates any new organization that is created with the following organization-level roles and permissions.

- **Organization Administrator.** Users in the Organization Administrator role can manage objects within a particular organization. This includes all child organizations of that organization. *This role cannot be modified or deleted*.
- **Asset Provider.** Users in the Asset Provider role can create new assets and view assets. This role also has the ability to register users to consume assets.
- **Asset Consumer.** Users in the Asset Consumer role can view all assets. This role also has the ability to register users to consume assets.
- **Asset Administrator.** Users in the Asset Administrator role can manage all assets within the organization.
- **Policy Administrator.** Users in the Policy Administrator role can manage design/ change-time and run-time policies within an organization.
- **API Runtime Provider.** Users in the API Runtime Provider role can manage run-time policies within an organization. This role also has the ability to configure run-time actions for virtual APIs.

- **API Publisher.** Users in the API Publisher role can publish run-time policies within an organization. This role also has the ability to publish APIs to gateways. For example, webMethods Mediator.
- **API-Portal Administrator.** Users in the API-Portal Administrator role can manage API-Portals within an organization.

Note: By default, the Asset Provider and Asset Consumer roles are automatically assigned to the Users group that CentraSite creates for an organization. Consequently, every user in an organization is assigned these roles. If you do not want every user in your organization to have these roles (for example, if you do not want every user to have the Asset Provider role), edit the role assignments for your organization's Users group.

Creating and Managing Roles

You can create custom roles in CentraSite as needed. Custom roles can contain system-level permissions and/or organization-level permissions. A custom role can contain organization-level permissions for multiple organizations (i.e., you can create a role that allows a user to manage the policies in two different organizations). You can also modify many of the predefined organizations that CentraSite installs.

Who Can Create and Manage Roles?

To create and manage (i.e., view, edit and delete) roles for an organization, you must belong to a role that has the Manage Users permission for the organization. Users in the Organization Administrator role have this permission, although an administrator can assign this permission to other roles.

Note: Users that belong to a role that includes the Manage Organizations permission have the Manage User permission by implication. Such users can create and mange groups in any organization to which their Manage Organizations permission applies.

Be aware that you when you define a role, you cannot assign permissions to the role that you do not have yourself. Therefore, if you belong to the Organization Administrator role (and you have no additional role assignments), you cannot create roles that have any additional permissions than the ones provided by the Organization Administrator role. For example, you could not create a role that included system-level permission or permissions for other organizations. A user with the Manage Organizations permission at the system-level (such as someone in the CentraSite Administrator role) would need to do that.

Creating Custom Roles

To create a custom role

1. In CentraSite Control, go to **Administration > Users > Roles**.
2. On the Roles page, click **Add Role**.

3. In the **Role Information** panel, specify the following fields:

In this field...	Do the following...
Name	Enter a name for the new role. A role name can contain any character (including spaces). **Note:** The role name must be unique within an organization.
Description	*Optional*. Enter a short description for the new role. This description appears when the user displays the list of roles in CentraSite Control.
Organization	Specify the organization to which this role belongs. (The drop-down list only displays organizations for which you have Manage Users permission.)

4. In the **Permissions** panel, click **Assign permissions**.
5. In the **Assign Permissions** dialog box, do the following
   a. Select the permission(s) that you want to assign to this role. (The list will only contain permissions that you are authorized to assign.)
   b. Click **OK**.
6. Click **Save** to create the new custom role in the CentraSite registry/repository.

Viewing and Editing Roles

You use the Edit Role page to examine and/or edit the properties of a role. When viewing or editing a role, keep the following points in mind:

- You cannot modify the CentraSite Administrator role.
- You cannot modify the Organization Administrator.

To view or edit the properties of a role

1. In CentraSite Control, go to **Administration > Users > Roles**.
2. By default, all of the available roles are displayed.

   If you want to filter the list to see just a subset of the available roles, type a partial string in the **Search** field. CentraSite applies the filter to the **Name** column. The **Search** field is a type-ahead field, so as soon as you enter any characters, the display will be updated to show only those roles whose name contains the specified characters. The wildcard character `%` is supported.
3. Locate the role that you want to view or edit.
4. From the role's context menu, select the **Details** command.

5. Examine or modify the attributes on the Edit Role page as necessary.

Field	Description
Name	The name of the role. A role name can contain any character (including spaces). **Note:** The role name must be unique within an organization.
Description	Additional comments or descriptive information about the role.
Organization	*Read-only.* The organization to which the role belongs.
Permissions	The settings on this profile identify the permissions that are assigned to this role.

6. If you want to add or remove permissions to/from the role, select the **Permissions** profile and do the following:
   a. To add permissions to the role, click **Assign Permissions** and select the permission that you want to add.
   b. To remove permissions from the role, select the permissions that you want to remove and click **Remove**.
7. If you have edited the settings on the Edit Role page, click **Save** to save the updated role.

Deleting Roles

You use the Roles page to delete the custom roles. When deleting a role, keep in mind that you cannot delete the CentraSite Administrator role or the Organization Administrator role (not even if you belong to the CentraSite Administrator role).

How to delete a role

To delete a role

1. In the CentraSite Control, go to **Administration > Users > Roles** to display the roles list.
2. Enable the checkbox next to the name of the role that you want to delete.
3. Click **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

How to delete multiple roles in a single operation

You can delete multiple roles in a single step. The rules described above for deleting a single role apply also when deleting multiple roles.

Important: If you have selected several roles where one or more of them are predefined roles, you can use the **Delete** button to delete the roles. However, as you are not allowed to delete predefined roles, only roles you have permission for will be deleted. The same applies to any other roles for which you do not have the required permission.

To delete multiple roles in a single operation

1. In CentraSite Control, go to **Administration > Users > Roles** to display the roles list.
2. Mark the checkboxes of the roles that you want to delete.
3. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Changing Passwords of Predefined Users and Login Users

CentraSite deals with two types of users:

- **Predefined users.** These users are used for internal communication between the various components of CentraSite, and also for guest access to CentraSite.
- **Login users.** These users represent real users who are defined in the user repositories that CentraSite uses for authentication of users. Login users can log in to CentraSite's graphical UIs.

There can be several Application Server Tiers (ASTs) accessing a single CentraSite registry, and any password change that occurs on one AST should be made known to the other ASTs.

You can change passwords of predefined users and login users using the command line tool CentraSiteCommand.cmd (Windows) or CentraSiteCommand.sh (UNIX) of Command Central. The tool is located in <*CentraSiteInstallDir*> /utilities.

If you start the command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

Changing the Password of a Predefined User

The following users are predefined in CentraSite:

- **DefaultUser.** This is the owner of all predefined objects. The predefined password for this user is `PwdFor_CS21`. You should change this password as soon as possible after you have installed CentraSite.

- **guest.** This user is configured to have access to only some resources, and for those resources to have only read access. The predefined password for this user is `guest`.

  If you wish to protect all data in the CentraSite Registry/Repository from read accesses from guest users you can change the password of this user.

- **UDDIsubscriptionUser.** This user is used for communication between the application server and the CentraSite UDDI server. The predefined password for this user is `UDDI4CentraSite`.

- **PurgeUser.** This is a user who can purge log records. The predefined password for this user is `LogPurger4CS`.

- **EventsUser.** The CentraSite Events Listener will use this user name for authentication before persisting event data to the RuntimeEvents Collection database. The predefined password for this user is `EventsManager4CS`. You can change this password or, alternatively you can change the "EventsUser" to a login user by configuring the Event Receiver.

Caution: You cannot log on to CentraSite Control using the user ID/password combination for any of these predefined users. Guest users can log on to CentraSite Control without a password by using the **Browse as Guest** link on the login page.

If you wish to change the password of a predefined user, use a command of the following format:

```
CentrasiteCommand set Password [-url <CENTRASITE-URL>] -user <USER-ID>
-password <PASSWORD> -predefinedUser <PREDEFINED-USER>
-newPassword <NEW-PASSWORD>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set Password` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `–user`.
`-predefinedUser`	The user ID of the predefined user whose password you wish to change.

Parameter	Description
`-newPassword`	The new password for the predefined user.

Example

```
CentrasiteCommand set Password
-url http://localhost:53307/Centrasite/Centrasite -user AdminUser -password ABCXYZ123
-predefinedUser DefaultUser -newPassword MyPassword2
```

Note: When the password for a predefined user has been changed, any application using this user needs to be adapted to use the new password.

Changing a Login User's Password in the Password Store

CentraSite provides a secure password store for managing the passwords of login users whose credentials are required for internal communication between CentraSite components.

Examples of such scenarios that require authentication credentials for internal communication are the use of policies such as Promote Asset and Initiate Approval. These policies cause a lifecycle model state change that requires the approval of an authorized login user.

This password store exists in parallel to the user repository that CentraSite uses for authentication of users. There is no automatic synchronization of passwords between the user repository and the password store. The password for a given login user in the password store must be the same as the password for the same login user in the user repository. If you change a password in the user repository, you must manually update the password in the password store to the same new password.

The password store resides on the Software AG Runtime. If your CentraSite configuration uses more than one application server tier (AST), you must ensure that each AST uses an up-to-date version of the password store.

If you wish to change the password of a login user in the password store, use a command of the following format:

```
CentrasiteCommand set Password [-url <CENTRASITE-URL>] -user <USER-ID>
-password <PASSWORD> -userToStore <USER-TO-STORE>
-passwordToStore <PASSWORD-TO-STORE>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set Password` utility:

Parameter	Description
-url	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
-user	The user ID of a user who has the CentraSite Administrator role.
-password	The password of the user identified by the parameter -user.
-userToStore	The user ID of the login user whose password you wish to change in the password store.
-passwordToStore	The new password to be stored in the password store for the login user identified by the parameter -userToStore.

4 Importing/Exporting Registry Objects

Introduction

CentraSite enables you to export registry objects from a registry to an archive file on the file system, and import them from the archive file into the same registry or to another instance of CentraSite.

Please note that the import/export feature is designed specifically for exporting selected objects from the registry. If you want to make a copy of an entire instance of the CentraSite registry/repository in order to move it to another machine or for disaster recovery purposes, see "Backing Up the Database" on page 27.

Important: Registry objects should only be imported and exported between CentraSite versions of the same release level; the import/export feature is *not* intended for migration purposes.

Which Objects Can You Export and Import?

The following native CentraSite objects are objects that you can select for export:

- Organizations

  Note: By exporting an organization, it is possible to export and import the associated users, groups, roles and permissions.

- Users, including the groups that contain the users
- Asset Types (definitions) and Virtual Types
- Taxonomies
- Lifecycle Models
- Design-Time Policies
- Assets (instances)
- Run-Time Policies
- Report Templates

In addition, you can select the following asset types belonging to other components of the webMethods Product Suite for export:

- BPM Process Project
- IS Package (from the Integration Server).

Overview of the Export/Import Process

Exporting/importing registry objects and repository items is done in two steps; the exported objects are first copied to an *archive file* (a zip file that contains the exported registry objects and repository items) and at a later stage the contents of the archive file are imported into a target registry (which may be the registry from which the objects were exported or a different one).

Exporting an Object

You can export an object:

- Using the CentraSite Control user interface, as described in "Exporting and Importing Objects Using CentraSite Control" on page 158
- From the command line, as described in "Exporting and Importing Objects from the Command Line" on page 164

Be aware that when you export an object, CentraSite generally exports a number of additional objects besides the one you select. The specific set of objects that CentraSite exports with an object varies by object type (this topic is discussed in more detail below).

To export an object successfully, you must have view permission on the object that you select for export *and* all of the additional objects that CentraSite will export with that object.

After CentraSite completes the export process, it generates a report that identifies each of the objects that it attempted to export and indicates whether or not the object was exported successfully. The export report also identifies all referenced objects that were omitted from the archive file.

Important: If your site uses custom export handlers, consult your administrator for information about their behavior.

How the Export Process Handles Object Ownership

An object's ownership attributes are *included* with the exported object. However, this data is imported with an object only if the **Keep current owner** and **Keep current organization** options are enabled at import time. For more information about these options, see "How the Import Process Assigns the Ownership of an Imported Object" on page 151.

How the Export Process Handles Instance Level Permissions

The export process *does not* export the instance-level permissions that are assigned to an object.

Objects That CentraSite Includes When It Exports an Object

The additional objects that CentraSite includes in the archive file are determined by the specific export handler that is associated with the exported object's type. Generally speaking, CentraSite includes the following objects with any object that you export:

- The supporting documents that are attached to the object.
- The type definition of the object, if the object type is a custom object type or a modified predefined object type.
- The taxonomy associated with the object type, if the taxonomy is a custom taxonomy or a modified predefined taxonomy.

Additionally, if the object is an instance of a composite type, CentraSite also exports the object's components (shared and non-shared) and required objects. For a description of composite types, see "Object Type Management" on page 219.

For the exact set of objects that CentraSite exports for a given type of object, see "Exporting and Importing Specific Object Types" on page 153.

Objects That CentraSite Does Not Include When It Exports an Object

Normally, the export set for an object contains all of the objects referenced by the object (referenced standard objects such as predefined asset types are not included, since these can be expected to exist on the target machine). Nevertheless, you might want to examine the export report for warnings. If there are warnings and they identify objects that will not be present in the target registry at import time (usually indicated by a INMIEW0038 warning in the report), you must do one of the following:

- Export the referenced objects separately and import them into the target registry before you import the archive file that you just created.

  —OR—

- Use the **Add to List** command to build a list that includes the object that you want to export *and* all of the objects that are identified by the INMIEW0038 warnings. Then, export the list. Doing this will produce an archive file that contains the object that you want to export plus all of the objects that it references.

Exporting Objects That Use Custom Associations

CentraSite does not automatically export the custom association type definitions for objects that contain custom associations. If you export an object that is related to another object through a custom association, CentraSite will export the related object, but will not export the type object that defines the custom association. If that type definition does not exist on the target registry, you must import it beforehand or include it in the archive file with the object that contains the custom association.

How the Export Process Handles Supporting Documents

When you export an asset, all of the asset's supporting documents are exported with the asset.

Note: The export feature does not provide a way to export supporting documents or other repository items directly. However you can use the **Download** button on the Supporting Document Library page to download documents from one instance of CentraSite and then upload them to another.

How the Export Process Handles Other Versions of an Object

If you export an asset that is versioned, the exported asset will include a reference to the *previous version* of the asset. However, the referenced version itself will not be included in the archive. Unless the referenced version exists on the target machine or you create an archive that includes the referenced version of the asset, the reference is removed on import.

Exporting Objects Between Different Versions of CentraSite

CentraSite does not restrict the transfer of objects between different product versions. However, if for example a user-defined type is exported and already exists on the target machine, the import will fail if the existing type already has instances and the modification of the type due to the different product version is such that the existing instances do not validate with the type any longer. Typically, enhancements of the type with additional fields do not cause any problems, but modification of existing fields may lead to unexpected results.

Importing an Object

You import an object by importing the archive file to which it was previously exported. You can import an object into the same CentraSite registry from which it was originally exported or to a different CentraSite registry.

The import dialog displays the contents of the archive to be imported, and you can select either the entire archive or just a subset of the objects to import.

Note: If the archive contains a registry object with references which cannot be satisfied during import, the import process will continue but this object is not imported.

To import an object, you must have permissions to create that type of object in the target registry. If the object that is to be imported already exists in the target registry, you must have permission to edit the existing object. If you attempt to import an object but do not have the proper permissions, the import process will fail.

To import object types, you should have the role Asset Type Administrator. To import organizations or users, you must have the role *CentraSite Administrator*. Without that role, the importing of user-defined types will fail.

When an archive is imported, the importer reads the contents of the archive file and either adds its contents to the target registry (if the object does not already exist), or replaces existing objects with the objects from the archive. The importer checks the object's UUID to determine whether it already exists.

An object is ignored when the same object with an identical timestamp already exists. If objects are identical but the object to be imported has an older timestamp than the one in the registry, the import is rejected.

In the import dialog you can use the option **Assign new organization** to select a new organization into which the objects will be imported, or you can retain the original organization, in which case the original associations of the assets are maintained if the organization is available on the target; if the organization does not exist on the target, the import will fail and the importer log will record this fact.

In some cases, the original organization will be preserved during import even if you have selected a specific organization in this field. This happens if the object to be imported is any of the following:

- an organization
- a system-wide lifecycle model
- a system-wide policy

Asset types are not owned by any organization, so selecting an organization for such objects has no effect.

If you set the **Allow replace of registry objects** option in the import dialog, the imported object will overwrite the existing object even if it is older than the object in the target registry.

If you set the option **Keep current owner** in the import dialog and the original owner exists on the target machine, the assets will belong to that user after import; as a further requirement, this user on the target machine must have the same UUID as the original owner. If this user does not exist on the target, the import will fail and the importer log will record this fact. The **Keep current owner** option will be disregarded if a conflicting user is ignored at import; in that case the import will succeed and the objects will belong to the importing user.

Understanding Object Identity

Each registry object is identified by its own unique Universally Unique Identifier (UUID). CentraSite assigns a UUID to an object when it adds the object to the registry. After an object is created, its UUID cannot be changed.

You will sometimes see an object's UUID displayed in the user interface. When the UUID is shown for an object, it appears in the object's `Key` attribute, and it looks something like this: uddi:2d621948-89f3-11df-851c-a3fb7c9c098e.

During an import/export process the importer uses the UUID to determine whether an imported object already exists in the target registry. If an identical object (i.e., an object with an identical UUID) already exists in the target registry, the timestamps of both the existing object and that to be imported determine whether and in which way the importing process continues.

What Happens When an Imported Object Already Exists in the Target Registry?

When the user attempts to import an archive and the importer determines that an object in the archive file already exists in the registry, there are different ways for the importer to react, depending on the timestamps of the objects:

- The imported object has a newer timestamp than the existing one. The existing object is *replaced* and the import executes successfully.
- The timestamps are identical. The object is *ignored* and the import executes successfully.
- The imported object has an older timestamp than the existing one. The object is *rejected*, but this does not cause the import process to fail.

  If the **Allow replace of registry objects** option has been set in the import dialog, the existing object will be overwritten automatically even if the imported object is older than the object in the target registry. This may be a potential source of problems, because all objects in the registry depending on it will be affected as well.

  If you intend to use this functionality to transfer objects between instances of CentraSite, make sure that the system clocks on all of the involved servers are synchronized.

How the Import Process Assigns the Ownership of an Imported Object

The organizational and user ownership information for an exported object is present in the archive file. However, how ownership is assigned to the imported object is determined by 1) whether the object is added to the target registry as a new object or replaces an existing object and 2) whether the **Keep current owner** option is set.

- If the imported object replaces an existing object in the registry, the existing object is replaced, but its **Owner** and **Organization** attributes remain unchanged (i.e., the updated object belongs to the same organization and user as it did before it was replaced by the copy from the archive file).
- If the imported object is added to the registry as a new object, its organizational and user ownership is determined by the following parameters:
  - If **Keep current owner** is set in the import dialog, and the original owner exists with the same UUID on the target, the objects will belong to that user after import; if the user with the same UUID does not exist on the target, the import fails.

    If **Keep current owner** is *not* set in the import dialog, the importing user will be the owner of the object.
  - If **Keep current organization** is set in the import dialog, the original associations of the objects are maintained if the organization exists with the same UUID on the target; if the organization with the same UUID does not exist on the target, the import will fail.

How Referenced Objects Are Handled During Import

If an object to be imported contains a reference to another object, the importer determines whether the referenced object is contained in the archive file or present in the registry. If the referenced object is present in the archive file, the importer will import it as necessary to resolve the reference.

If the referenced object is not present in the archive file or the target registry, the object containing the reference is not imported.

How Lifecycle State Is Handled During Import

If an object to be imported includes state information, the importer checks whether a lifecycle model containing the specified state exists on the target registry.

The option **Keep lifecycle state** in the import dialog determines whether the lifecycle state of assets in the archive should be preserved during the import. If you mark the checkbox, the lifecycle state of each imported asset will be set to the same value as the asset's lifecycle state in the archive being imported.

This operation is available when the lifecycle model itself is in the `productive` or `retired` state.

This operation is only possible for any given asset if the lifecycle model governing the imported asset contains the same lifecycle state as the state in which the asset was originally exported to the archive. If the lifecycle model for the imported asset does not contain the same state as the state of the asset in the archive, the state of the imported asset will be set to the initial state of the lifecycle model that governs the imported asset.

If the target registry has no lifecycle model for the type of object that you are importing, the imported object's lifecycle state information, if present in the archive file, is ignored.

If the target registry uses a different lifecycle model than the one used by the imported object, the object's lifecycle state information, if present in the archive file, is ignored and the object enters the initial state of the lifecycle model that is in effect for its type on the target registry.

Important: If the object you are importing was exported from an instance of CentraSite that has assigned a "stage" to the object's lifecycle state, the object can only be imported to the registry whose address is specified in that stage.

What Policies Are Triggered During an Import

When the import process adds a new object to the registry, CentraSite will apply relevant PreCreate and PostCreate policies in the same way as if you had created the object manually.

If the import process replaces an existing object in the registry, CentraSite will likewise trigger the applicable PreUpdate and PostUpdate policies.

Note, however, that these policies are *not* activated when a complete organization is imported.

For more information about policies, see the *CentraSite User's Guide*.

Promotion

CentraSite supports configurations where different CentraSite instances are used to represent lifecycle stages or usages. So CentraSite could for example be set up with two instances **development** and **production** to represent the different phases in a software development lifecycle; it could also be set up with instances differentiating between asset creation and asset consumption aspects.

Promotion refers to the capability to copy an asset, a lifecycle model or a policy from one lifecycle stage to another; this is done by exporting the object from its current registry and importing it into the registry that hosts the next phase of its lifecycle.

A stage definition in CentraSite is managed by a lifecycle stage object which describes a CentraSite instance by name and configuration information.

Those stages are assigned to lifecycle model states to define the allowed promotion paths for objects in a certain lifecycle state. To promote an object from one lifecycle stage to another, the object is exported (and optionally deleted) from the source registry and then imported in the target registry.

When any object from the source registry is in an end state, one or more registries can be defined to which the object can be imported; attempting to import it into any other target will subsequently fail.

On import, the registry object keeps its lifecycle state if its lifecycle model is available in the target registry; if the model does not exist, the object is set to the initial state in the default lifecycle model.

Exporting and Importing Specific Object Types

Important: If your site has installed custom export handlers for certain object types, consult your administrator for information about the export sets that they produce.

Organizations

When you export an Organization, the export set consists of:

- The definition of the organization itself
- The contents of the Supporting Documents library of the organization

Additionally, you can optionally export the following objects with the organization:

- Lifecycle models
- Policies

- Child organizations with their related objects
- Assets belonging to the organization
- Definitions of users, groups and roles, together with their associated permissions.

Note: Keep the following in mind:

- Groups, Roles and Permissions objects cannot be directly exported. You can only export these kinds of objects by exporting the organization to which they belong.
- If lifecycle models and policies are selected to be exported with the organization, users are automatically selected for export also. If lifecycle models and policies are not included in the export, you have the option to select the users for export.
- If the organization is exported with users and a user is a member of one of the organization's groups without being related to the organization itself, then this user will not be exported with the organization. This is true even if the user is the Organization Administrator.
- If the organization was exported without users or if the Organization Administrator belongs to a foreign organization, the Organization Administrator will be set to the importing user on import; otherwise the Organization Administrator will not change. An organization's Primary Contact will always be the importing user.
- If the replace option was set and the imported organization is already present in the target registry, its associated groups and users will be updated. Furthermore, if the export included users, the groups are entirely replaced with the corresponding information of the export set, i.e., users may be added to or deleted from the organization and the groups. However, if the export did not include users, the organization and its groups will retain their users.
- If an archive with one or more organizations is imported, then the **Keep current organization** option is implicitly switched on, thereby ensuring that all objects will be imported to their original organization.

Users

When you export a user, the export set consists of:

- The user object itself.
- All groups that contain the user.

If an organization with users is imported where the user is already present in the target registry, the import of the user object is ignored and the object is not replaced. This is also true when only the user name but not its `Key` attribute are identical; in this case the conflicting user will be ignored at import and a warning message is displayed. The user is removed from its group and any **Keep current owner** settings referring to this user are ignored.

If a user was not already present in the target registry, the organization of the user in the target registry is decided as follows:

- If "Keep Organization" is selected during the import and the corresponding organization is not available in the target registry, then the import fails.

- If you select a new organization during the import, then the user is assigned to the selected organization in the target registry.
- The same rules apply for imported groups.

Asset Types and Virtual Types

When you export an asset type or a virtual type, the export set consists of:

- The type definition.
- The XSD file that contains the type definition.
- Optionally, all instances of the asset type or virtual type.
- The user defined or customized predefined type definitions of all types and association types that are referred to by the asset type or virtual type being exported. This applies recursively, i.e., if a referenced asset type also contains such references, then these are also included in the export set. The instances of such referenced types are not included in the export set, even if the option **Include instances** is set.
- Other user defined or customized predefined objects (taxonomies etc.) that are referred to by the type being exported.

Taxonomies

When you export a taxonomy, the export set consists of:

- The taxonomy object
- All categories belonging to the taxonomy (the entire dependency tree)
- Repository resources (icons)
- Optionally, all objects classified by categories of this taxonomy.

Lifecycle Models

When you export a lifecycle model, the export set consists of:

- The lifecycle model object itself.
- Optionally, all objects governed by the lifecycle model.
- All references to lifecycle state permissions. During the import of the archive in the target registry, if the user/group referenced in the lifecycle state permission exists in the target registry, then the permissions will be saved, otherwise the references will be removed during the import.

Design/Change-Time Policies

When you export a design-time policy, the export set consists of:

- The design/change time policy
- Policy actions that the policy uses

For detailed information about exporting and importing design/change-time policies, see the *CentraSite User's Guide*.

Asset Instances

When you export an instance of an asset, the export set consists of:

- The asset object itself.
- All supporting documents associated with the asset.
- The asset's type definition, if the type is a custom type or a modified predefined type.
- The taxonomy associated with the asset type, if the taxonomy is a custom taxonomy or a modified predefined taxonomy.
- All referenced objects to which the asset has an association.
- For each exported asset instance, the export set contains all referenced asset types and association types. This means that when the export set is imported on the target machine, imported assets have no unsatisfied references.
- Assets that are referenced by the assets that you wish to export (if you select the option **Include assets referenced by selected assets**). If referenced assets themselves reference other assets, also those assets will be included; this selection process is repeated until all asset references are satisfied.

  Note: This option can cause the size of the export set to be very large.

Note: When you export a service asset that refers to XML Schemas, the referenced XML Schemas are also exported automatically, provided that you have permission to export them. If you do not have permission to export a referenced XML Schema, the referenced XML Schema will not be exported and a warning message will be logged.

If the asset is an instance of a composite asset, the export set will also include:

- All components (shared and non-shared) associated with the asset.
- All required objects associated with the asset.

For the list of shared and non-shared components associated with the predefined composite types installed with CentraSite, see "Working with Composite Types" on page 269.

Report Templates

When you export a report template, the export set consists of:

- The report template itself.
- The associated RPT design file from the repository.

BPM Process Project

The asset type `BPM Process Project` is one of the webMethods Product Suite asset types for which CentraSiteprovides an export function.

When you export an instance of a BPM Process Project, the export set consists of:

- The BPM Process Project asset itself.
- The BPM Process Project's type definition, if the type is a custom type or a modified predefined type.
- All components (shared and non-shared) associated with the asset.

  For the list of shared and non-shared components associated with the predefined composite types installed with CentraSite, see "Working with Composite Types" on page 269.
- All required objects associated with the asset.

Integration Server Assets

The asset type `IS Package` is one of the webMethods Product Suite asset types for which CentraSiteprovides an export function.

When you export an instance of an IS Package, the export set consists of:

- The asset itself.
- The asset's type definition, if the type is a custom type or a modified predefined type.
- All components (shared and non-shared) associated with the asset.

  For the list of shared and non-shared components associated with the predefined composite types installed with CentraSite, see "Working with Composite Types" on page 269.
- IS Server assets that are referenced by the assets that you wish to export (if you select the option **Include Assets referenced by selected Assets**). If referenced assets themselves reference other assets, also those assets will be included; this selection process is repeated until all asset references are satisfied.

Exporting and Importing Objects Using CentraSite Control

Exporting Objects Using CentraSite Control

With CentraSite Control, you can export one or more objects in each export operation. To export an object, you typically select it in the navigation view and then specify which related objects you would like to export along with it.

The export operation creates an archive file on the file system. The archive file contains a copy of the objects that you have exported. The archive file can be imported afterwards into the same CentraSite registry or into a new registry. For details about the import operations, see "Importing Objects from an Archive Using CentraSite Control" on page 161.

Objects can be exported either by right-clicking the object you wish to export and choosing **Export** on the context menu, *or* by using the **Actions > Export** menu entry, *or* by clicking the **Export** icon. The **Actions > Export** menu entry and the **Export** icon allow you to export multiple objects in one step whereas the context menu only allows you to export a single object.

If you have created a list in your **My Favorites** area, you can export the set of objects in the list by using the **Export** option in the list's context menu. For information about defining lists, see *Getting Started with CentraSite*.

Some objects in the registry do not support all of the export methods; check the user interface to see which controls are available for the type of object you want to export.

Note: You must have *view* permission for the assets you want to include in the export set.

To export an object or a set of objects

1. In CentraSite Control, go to the page that contains the object or the set of objects that you want to export. For example, if you want to export a taxonomy, go to the **Administration > Taxonomies** page, or if you want to export the contents of a list, go to **Home > My CentraSite > My Favorites** to see your defined lists.
2. Locate the object or the set of objects you want to export and choose **Export** from the context menu,

   -OR-

   If the **Actions** menu is visible, select the object or set of objects you want to export and choose the **Actions > Export** menu entry.

   -OR-

   If the **Export** icon is visible, select the object or set of objects you want to export and click the **Export** icon.

3. The export dialog shows the selected objects and all dependent objects. The colored icon beside each object indicates whether the object is one of the selected objects or a dependent object.

   The checkbox beside each object indicates whether or not the object should be included in the export set. By default, all displayed objects are included in the export set.

   If you wish to remove an object from the export set, unmark its checkbox. This removes the object and all of its dependent objects (if any) from the export set.

4. For some object types, additional export options are available. In this case, the button **Export Options** is activated, and when you click this button, a dialog opens that shows these additional export options. If you are exporting objects that are contained in a list, and the list contains more than one object type, you will see several tabs, with one tab per object type.

   The available options depend on the type of object you wish to export.

Object type	Available export options
Asset	■ **Include assets referenced by selected assets.** If the selected assets contain references to other assets, then include the referenced assets also in the export set. This selection process is repeated recursively until all asset references are satisfied.
Lifecycle Model	■ **Include assets for selected lifecycle models.** Include all assets that are governed by the lifecycle model. ■ **Include assets referenced by selected assets.** If the assets that are selected by the option **Include assets for selected lifecycle models** contain references to other assets, then include the referenced assets also in the export set. This selection process is repeated recursively until all asset references are satisfied.
Taxonomy	■ **Include assets for selected categories.** Include all assets that are classified by categories of the taxonomy. ■ **Include assets referenced by selected assets.** If the assets that are selected by the option **Include assets for selected categories** contain references to other assets, then include the referenced assets also in the export set. This selection process is repeated recursively until all asset references are satisfied.
Asset Type	■ **Include assets for selected asset types.** Include all assets that are instances of the asset types.

Object type	Available export options
	■ **Include assets referenced by selected assets.** If the assets that are selected by the option **Include assets for selected asset types** contain references to other assets, then include the referenced assets also in the export set. This selection process is repeated recursively until all asset references are satisfied.
Organization	■ **Include users.** Include all users who belong to the organization. ■ **Include child organizations.** Include all organizations that are child organizations of the organization. If the option `Include assets of organization` is selected, include also all assets that belong to the child organizations. ■ **Include lifecycle models and policies.** Include all lifecycle models and policies that have been define for the organization. ■ **Include assets of organization.** Include all assets that belong to the organization. ■ **Include assets referenced by selected assets.** If the assets that are selected by the option **Include assets of organization** contain references to other assets, then include the referenced assets also in the export set. This selection process is repeated recursively until all asset references are satisfied.

Click **OK** to complete the object selection.

5. Click **OK** to start the export.
6. Specify a name for the archive file when prompted to do so.
7. When the export is complete, examine the export report to verify that the objects were exported successfully.

   A confirmation message appears when the export is complete. Here, you have the possibility to view the export log, in order to verify that the objects were exported successfully. To do this, click on the **Export Log** link.

   Important: If the report indicates that certain associated objects have been omitted from the archive, you will need to make sure that these objects are either present in the target registry when the archive is imported or create an archive that includes them. For additional information about exporting associated objects, see "How the Export Process Handles Object Ownership" on page 147.

Importing Objects from an Archive Using CentraSite Control

You can import objects from an archive file that was previously created by using the CentraSite export feature. The archive you wish to import must reside in the file system of the computer where your browser is running.

For general information about what happens during the import process, see "Importing an Object" on page 149.

To import objects from an archive file

1. In CentraSite Control, go to any page that displays the **Import** button. Examples of pages that include this button are:

   Asset Catalog > Browse

   Policies > Design/Change-Time

   Administration > Taxonomies

2. Click the **Import** button. This activates the import dialog.

   In the field **Import as**, select **Archive** from the drop-down list. When you select this option, the layout of the wizard changes to show just the fields that are required for importing an archive.

   In the field **File**, supply the name of the file that contains the archive.

   Click **Finish**. The **Import Preview** page is now displayed.

   Note: If the archive you wish to import was created with CentraSite 8.2 or earlier, the **Import Preview** page is not available. In this case, when you click **Finish**, the import operation continues with the **Import Options** dialog in step 4.

3. The **Import Preview** page displays the names of the top-level objects contained in the export archive. If the archive contains related objects that the top-level objects require for completeness, the related objects are not displayed, but they will be imported automatically along with the top-level objects. For example, an exported web service requires a related schema, so the archive file will contain both the web service and the schema, and an import of the web service will cause the related schema to be imported also.

   By default, all displayed top-level objects are selected for import; this is indicated by the marked checkbox beside the name of each object.

   You can exclude objects from the import by unmarking their checkboxes.

   Note: In some cases, there may be a dependency between top-level objects (for example, a web service that refers to a taxonomy), and the import operation ensures that such dependencies are retained. This means that if you unmark the checkbox of a top-level object that is required by another top-level object, CentraSite will ensure that the required object is nevertheless included in the import.

Additional options are available that allow you to choose how the imported objects will be created. To access these options, click **Import Options**.

4. In the **Import Options** dialog, set the following options.

Option	Meaning
Keep current organization / Assign new organization	When you import objects, you can import them into the same organization in the target registry as in the source registry from which they were exported, or you can assign a new owning organization. Choose **Keep current organization** to import the objects into the same organization. The organization in the target registry must have the same name and UUID as in the source registry. Choose **Assign new organization** to import the objects into a new organization. If you choose this option, you can select the new organization via the **Select Organization** button. In some cases, the original organization will be preserved during import even if you have selected a specific organization in this field. This happens if the object to be imported is any of the following: ■ an organization ■ a system-wide lifecycle model ■ a system-wide policy Asset types are not owned by any organization, so selecting an organization for such objects has no effect.
Keep current owner / Assign new owner	The imported objects can be assigned to the same owner as in the source registry, or you can assign a new owner. Choose **Keep current owner** to assign the imported objects to the same owner as in the source registry. The owner in the target registry must have the same name and UUID as in the source registry. Choose **Assign new owner** to assign the imported objects to a new owner. If you choose this option, you can select the new owner via the **Select Owner** button.

Option	Meaning
Keep lifecycle state	This option determines whether the lifecycle state of the imported assets is preserved. Enable the option to retain the lifecycle state of the assets being imported. For more information, see "How Lifecycle State Is Handled During Import" on page 152.
Allow replace of registry objects	Choose this option to specify that existing objects with the same name in the target registry will be overwritten, even if the object in the archive is older than the one in the target registry. For more information about this option, see "What Happens When an Imported Object Already Exists in the Target Registry?" on page 151.
Import groups that the user belongs to	When you import a user, you can specify whether you want to also import the groups that the user belongs to. This applies only to user-defined groups; system-defined groups are not imported. Choose this option to import the user-defined groups that the user belongs to.

5. Click **OK** to leave the **Import Options** dialog.

   If you are importing an archive that was created with CentraSite 8.2 or earlier, the import now starts.

   If you are importing an archive that was created after CentraSite 8.2, the **Import Preview** dialog is displayed again. Now click **OK** to start the import.

6. When the import operation completes, the import wizard informs you if the import was successful or if there were any errors.

   You can click **OK** here to terminate the import wizard without viewing the import log.

   Alternatively, to see details of the objects that were imported, the wizard offers you a link to view the import log. When you click this link, the import log lists each top-level object and indicates whether or not it was successfully imported. The import log also lists the import status of any related objects that were contained in the archive (i.e. objects in the archive that are not top-level objects).

   In the import log view page, click **OK** to terminate the import wizard.

Hints for Importing

Handling of Missing Required Attributes of Assets When Importing

It is possible that an asset type definition in the source registry is different from an asset type definition with the same name in the target registry. This could happen if, for example, the asset type in the target registry represents an updated version of the asset type with different attribute definitions.

If an asset type has an attribute that is required in the target registry but not in the source registry, such a mismatch will occur.

The way CentraSite handles this situation on the target registry depends on the attribute type:

- If the attribute type is one that has a default value, such as `slot` or `classification`, CentraSite assigns the default value to the attribute when the asset is imported.
- If the attribute type is one that does not have a default value, for example an attribute type that represents a file name or relationship, an error will occur during the import of the archive, because the target registry requires a value for the attribute. Under such circumstances, the affected asset will not be imported and the import log will contain appropriate error messages. In such cases, you need to ensure that a value is supplied for the attribute before the asset is exported from the source registry.

Exporting and Importing Objects from the Command Line

Registry objects can be exported from and imported into CentraSite by using a command line interface. In either case (exporting and importing), you must include the CentraSite JAR files in your Java classpath. It is usually convenient to include all the JAR files that are in the CentraSiteredist folder (typically C:\SoftwareAG\CentraSite\redist).

Exporting Objects from the Command Line

The syntax of the export call is as follows:

```
com.centrasite.importexport.ExportOperation [-noinstances] [-deleteafterexport]
[-orgnorelated] [-orgwithpolicyandlifecycle] [-orgnochildorganization]
<CentraSite_URL> <archive_filename> <user> <password> { <concept> | <guid> }...
```

The meanings of the parameters are as follows:

Parameter	Description
`-noinstances`	When exporting a CentraSite asset type (represented by a Concept object) or a taxonomy (represented by a Classification

Parameter	Description
	Scheme object), no instances of them are exported. For other objects, this option is ignored.
`-orgnorelated`	When exporting an organization, related registry objects, that is users, groups, roles and permissions, will not be exported.
`-orgwithpolicyandlifecycle`	When exporting an organization, its related policies and lifecycles will also be exported.
`-orgnochildorganization`	When exporting an organization, its child organizations will not be exported.
`<CentraSite_URL>`	The URL of the CentraSite registry/repository; typically `http://localhost:53307/CentraSite/CentraSite`.
`<archive_filename>`	The name of the export archive file (ZIP file).
`<user>`	The user-ID of the CentraSite user to be used for the import operation.
`<password>`	The user's password.
`<concept>`	The name of a JAXR-based concept to be exported.
`<guid>`	The GUID of a CentraSite registry object to be exported, prefixed by `uddi:`, for example: `uddi:207ff1cc-25c5-544c-415c-5d98ea91060c`.

Importing Objects from an Archive Using the Command Line

The syntax of the import call is as follows:

```
com.centrasite.importexport.ImportOperation [-setreplace]
[-importorg <org_key>] [-keepowner] [-keeporganization] <CentraSite_URL>
<archive_filename> <user> <password>
```

The meanings of the parameters are as follows:

Parameter	Description
`-setreplace`	When an object is to be imported, the timestamp of the object in the import archive file is compared with the timestamp of the corresponding object with the same GUID in the registry, if the latter exists. If the timestamps are equal, the object is not imported. If the archive timestamp is older than the registry timestamp, the object is only imported if this optional parameter is present; otherwise, it is skipped.
`-importorg <org_key>`	By default, objects are imported into the current user's organization. Use this optional parameter if you want to specify a different organization into which the objects should be imported. The keyword is followed by the organization's GUID, prefixed by `uddi:`, for example: `uddi:207ff1cc-25c5-544c-415c-5d98ea91060c`. Note that this parameter has no effect if the `-keeporganization` parameter is set.
`-keepowner`	This option will maintain the original owner of the importing objects. This requires the original owner to exist with the same UUID on the target machine. If this parameter is not set, the importing user will be owner of the objects.
`-keeporganization`	This option will maintain the original organization of the importing objects. This requires the original organization to exist with the same UUID on the target machine. **Note:** If an organization is imported with its assets, this option is activated automatically.
`<CentraSite_URL>`	The URL of the CentraSite registry/repository; typically `http://localhost:53307/CentraSite/CentraSite`.
`<archive_filename>`	The name of the export archive file. The archive file can contain an organization with its assets or can contain a set of objects that were exported from one or more organizations.

Parameter	Description
`<user>`	The user-ID of the CentraSite user to be used for the import operation.
`<password>`	The user's password.

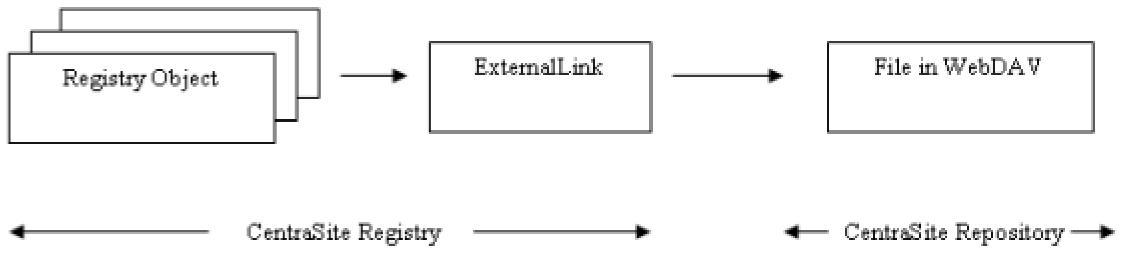
5 Importing Objects Using the API

Overview of Importing Objects Using the API

CentraSite has a set of importers for importing various kinds of objects. They are used by the Import function of CentraSite Control, but they can also be invoked from Java programs, through the SOAP API, from batch commands (e.g. shell scripts) or directly from the command-line prompt. Import APIs are available for importing:

- Web services
- XML schemas
- XML services
- REST services
- BPEL process files
- XPDL files

All the importers have a few characteristics in common. Each import function is called with an appropriate XML file (for example, the web service importer expects a WSDL file as input). The XML file can be located in the file system, or, for some importers, it can alternatively be specified by a URL (HTTP). The file is incorporated into the CentraSite repository, and there will be a registry object (an ExternalLink) that will point to it. Each importer typically creates one or more JAXR-based registry objects. The following picture illustrates the relationship:



Invoking an Importer from a Java Program

The classes used by the importers are contained in the file CentraSiteUtils.jar. The `CLASSPATH` variable must refer to the JAR files that are used by CentraSite. It is convenient to include all JAR files contained in the redist folder of the CentraSite installation.

After using an instance of the JAXRAccessor class to establish a valid connection (with user and password) to CentraSite, one or more imports are possible. Finally, close the session by calling the close() method of JAXRAccessor.

Importing a Web Service

The basic input when importing a web service is a WSDL file describing one or more services. There are also some setter methods for additional parameters. The organization (per object or per name) must be set; other parameters are optional.

The following example demonstrates how to register a web service to CentraSite:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.webservice.WebServiceRegistrator;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String organizationName = "MyOrganization";
String wsdlFile = "c:/temp/MyService.wsdl";
String user = ...;
String password = ...;

JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
    WebServiceRegistrator wsr = new WebServiceRegistrator(wsdlFile, jaxr);
    wsr.setOrganization(organizationName);
    wsr.register();
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

The WebServiceRegistrator class provides the setter methods listed below. The setter methods must be called before calling the register() method.

setDescription

Sets a description text that will appear with the imported Service object.

Syntax

```
public void setDescription(String description)
```

Parameters

String description
The description text.

Usage Notes

Optional.

setOrganization

Sets the name of the organization under which the service should be registered.

Syntax

```
public void setOrganization(String organizationName)
```

Parameters

String organizationName
The name of the organization.

setOrganization

Sets the organization object under which the service should be registered.

Syntax

```
public void setOrganization(Organization organizationObject)
```

Parameters

Organization organizationObject
The organization object under which the service should be registered.

setPackageName

Sets the name of the registry package of which the service should be a member.

Syntax

```
public void setPackageName(String packageName)
```

Parameters

String packageName
The name of the registry package of which the service should be a member.

Usage Notes

Optional.

setPackageObject

Sets the registry package object of which the service should be a member.

Syntax

```
public void setPackageObject(RegistryPackage packageObject)
```

Parameters

RegistryPackage packageObject
The registry package object of which the service should be a member.

Usage Notes

Optional.

setUsedObject

Sets a registry object that will point to the imported service with a `Uses` association; in other words, the imported service will be used by the specified object.

Syntax

```
public void setUsedObject(RegistryObject usedObject)
```

Parameters

RegistryObject usedObject
The registry object that will point to the imported service.

Usage Notes

Optional.

setUsesObject

Sets a registry object that will be pointed to by the imported service with a `Uses` association; in other words, the imported service will use the specified object.

Syntax

```
public void setUsesObject(RegistryObject usesObject)
```

Parameters

RegistryObject usesObject
The registry object that will be pointed to by the imported service.

Usage Notes

Optional.

setWebDAVFolder

Changes the path of the folder where the WSDL file is stored in the CentraSite repository.

Syntax

```
public void setWebDAVFolder(String folderPath)
```

Parameters

String folderPath
The path of the folder where the WSDL file should be stored.

Default: /projects/WSDL.

Usage Notes

Optional.

Updating a Registered Web Service

To update a registered web service (for example, when the WSDL file has been modified or if you want to change the registry package that contains the web service), call the register() method with the modified WSDL file. For details, see "Importing a Web Service" on page 171.

Note: The key associated with a registered service comprises its organization, its name and the WSDL file's targetNamespace. If you modify one or more of these, the service is not updated; rather, a new registry entry is created.

Attaching a WSDL to a Registered Web Service

It is possible to attach a WSDL file to an existing web service. The code for attaching a WSDL file is similar to the code for importing a WSDL file. An example code snippet follows:

```
WebServiceRegistrator wr = new WebServiceRegistrator("attach.wsdl", jaxr);
wr.setAttachServiceID("uddi...");
wr.register();
```

where `"uddi..."` denotes the UDDI key of the service to which the WSDL file should be attached.

Note the following:

- The service can be a manually-created service.
- If the service has already been registered with a WSDL, then calling setAttachServiceID updates the registered information.
- If there are any services within the WSDL, they are registered as usual.
- If the service has a ServiceBinding that is not in the WSDL, it is retained.
- If the service has a ServiceBinding that is in the WSDL, it is updated. Operations are never deleted.

The WebServiceRegistrator class provides the setter method (in addition to the setter methods described under Importing a Web Service). The setter method must be called before calling the register() method.

setAttachServiceID

Sets a Service ID (`getKey().getId()`) for a WSDL-service attachment.

Syntax

```
public final void setAttachServiceID(java.lang.String attachServiceID)
```

Parameters

java.lang.String attachServiceID)
The ID of the service, in the form `"uddi:..."`.

Removing a Registered Web Service

Use the functions described below to remove a registered web service and all its resources, including all associated registry objects and the WSDL file in the CentraSite repository.

You must use an instance of the JAXRAccessor class to establish a CentraSite connection before calling the removeService or removeServiceByID function. Finally, close the session by calling the close() method of JAXRAccessor.

Also before calling the removeService or removeServiceByID function, you may call setter functions, which are also described below.

The following example demonstrates how to remove a web service from CentraSiteCentraSite:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.webservice.WebServiceAdministrator;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String wsdlFile = "c:/temp/MyService.wsdl";
String user = ...;
String password = ...;
 JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
   WebServiceAdministrator wsa = new WebServiceAdministrator(jaxr);
   int removeCount = wsa.removeServices(wsdlFile);
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

The WebServiceAdministrator class provides the methods shown below:

removeService

Removes the service specified by the unique name and namespace.

Syntax

```
public boolean removeService(String serviceName, String namespace)
```

Parameters

String serviceName
The name of the service to be removed.

String namespace
The namespace of the service to be removed.

Return Codes

Value	Meaning
true	The specified service was successfully found and removed.
false	Otherwise.

removeServices

Removes from CentraSite all services that are indicated by the specified WSDL file.

Syntax

```
public int removeServices(String wsdlFilename)
```

Parameters

String wsdlFilename
The name of the WSDL file that indicates the services to be removed.

Return Codes

Value	Meaning
int	The number of services that were removed.

removeServiceByID

Removes the service with the specified ID.

Syntax

```
public boolean removeServiceByID(String serviceID)
```

Parameters

String serviceID
The ID of the service to be removed. The ID is the getKey().getID() value of the associated service object.

Return Codes

Value	Meaning
true	The specified service was successfully found and removed.
false	Otherwise.

setDeleteTargetAssocs

Specifies whether or not to delete associations to the service object where the service object is the target of the association. If the parameter is `false` and the service object is the target of one or more associations, the removal of the service object is inhibited.

Syntax

```
public void setDeleteTargetAssocs(boolean deleteTargetAssocs)
```

Parameters

boolean deleteTargetAssocs
true: Delete associations to the service object where the service object is the target of the association.

Default: true.

setRemoveReferencedImports

Specifies whether or not to remove from the repository referenced imported WSDL and schema files, together with their controlling ExternalLinks.

Syntax

```
public void setRemoveReferencedImports(boolean removeReferencedImports)
```

Parameters

boolean removeReferencedImports
true: Remove the referenced imported WSDL and schema files.

setRemoveWsdl

Specifies whether or not the WSDL file together with its controlling ExternalLink should also be removed from the repository.

Syntax

```
public void setRemoveWsdl(boolean removeWsdl)
```

Parameters

boolean removeWsdl
true: Remove the WSDL file together with its controlling ExternalLink from the repository.

Default: true.

Finding a Registered Web Service

The Web Service API provides functions with which you can find a registered web service. The following example demonstrates this:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.webservice.WebServiceAdministrator;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String wsdlFile = "c:/temp/MyService.wsdl";
String user = ...;
String password = ...;

JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
   WebServiceAdministrator wsa = new WebServiceAdministrator(jaxr);
   Collection services = wsa.findWebServices(wsdlFile);  // Service object coll.
   . . .
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

The WebServiceAdministrator class provides the methods shown below:

findWebServiceByNamespace

Finds the registered web service on the basis of its name and namespace.

Syntax

```
public Service findWebServiceByNamespace(String serviceName, String namespace)
```

Parameters

String serviceName
The name of the registered web service.

String namespace
The namespace of the registered web service.

findWebServices

Finds a collection of web service objects that are registered with the specified organization in the specified WSDL file.

Syntax

```
public Collection findWebServices(String organizationName, String wsdlFile)
```

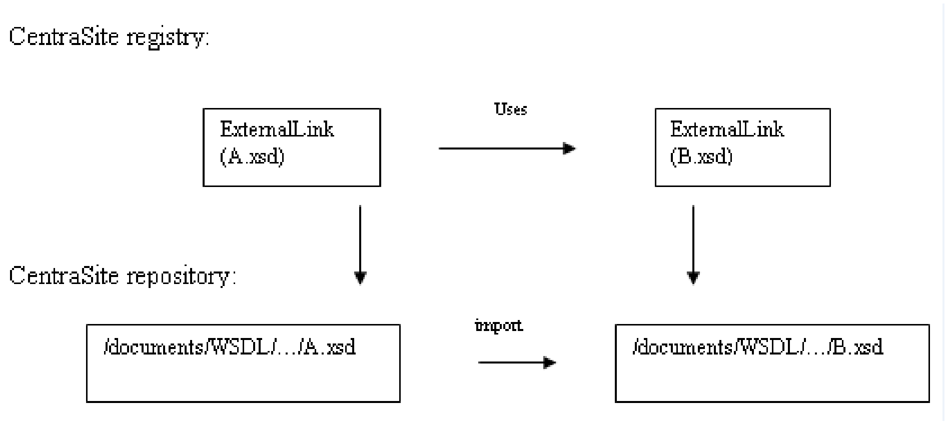
Parameters

String organizationName
The name of the organization.

String wsdlFile
The filepath to the WSDL file.

Importing a Schema

The schema importer works closely together with the web service importer. Since a WSDL file of a web service may import or include schema files, CentraSite enables you to design and store a schema before referencing WSDL files. Similarly, if a schema file is imported by more than one WSDL file, it could be beneficial to register the schema first, before registering the WSDL files. When a schema is imported, the file is copied into the CentraSite repository and an ExternalLink that controls the resource is created. If a schema imports (includes) further schemas, then the referenced schemas are also imported. The referenced relations are established by means of a `Uses` association in the registry. The following picture illustrates the relationship:



The following example demonstrates how to import a schema file into CentraSite:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.schema.SchemaImporter;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String xsdFile = "c:/temp/MySchema.xsd";
String user = ...;
String password = ...;

JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
   SchemaImporter si = new SchemaImporter(xsdFile, jaxr);
   si.add();
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

Removing a Schema

You can remove a schema from CentraSite. This function deletes the XML Schema object, the ExternalLink and the resource in the repository.

Note: If a schema is referenced by another object, for example if it is referenced by the ExternalLink of a WSDL object, then it cannot be deleted.

The following example demonstrates how to remove a schema from CentraSite:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.schema.SchemaAdministrator;
import javax.xml.registry.infomodel.ExternalLink;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String user = ...;
String password = ...;
ExternalLink schemaExtLink = ...;

JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
   SchemaAdministrator sa = new SchemaAdministrator(jaxr);
   sa.remove(schemaExtLink);
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

The SchemaAdministrator class provides the following methods for removing a schema:

remove

Removes the schema at the specified location in the CentraSite repository, which may be relative or absolute.

Syntax

```
public boolean remove(String schemaLocation)
```

Parameters

String schemaLocation
The repository location, which may be relative or absolute.

Return Codes

Value	Meaning
true	The specified schema was found and removed.

remove

Removes the schema specified by its XML schema object.

Syntax

```
public boolean remove(RegistryObject schemaObject)
```

Parameters

RegistryObject schemaObject
The XML schema object or the external link of the schema.

Return Codes

Value	Meaning
true	The specified schema was found and removed.

removeCascading

Removes the specified schema and also all schemas that are related to it by import and/ or include. The initial schema is specified by its location in the CentraSite repository, which may be relative or absolute.

Syntax

```
public boolean removeCascading(String schemaLocation)
```

Parameters

String schemaLocation
The repository location, which may be relative or absolute.

Return Codes

Value	Meaning
true	The specified schema and all related schemas were found and removed.

removeCascading

Removes the specified schema and also all schemas that are related to it by import and/or include. The initial schema is specified by its XML schema object or by its external link.

Syntax

```
public boolean removeCascading(RegistryObject schemaObject)
```

Parameters

RegistryObject schemaObject
The XML schema object or the external link of the schema.

Return Codes

Value	Meaning
true	The specified schema and all related schemas were found and removed.

Importing a BPEL Process File

The BPEL importer imports objects of a Business Process Execution Language file. In CentraSite there are various specific predefined BPEL-ObjectTypes. A BPEL process flow usually references certain web services. Note that those web services should be registered prior to the BPEL registration; otherwise the references to the services cannot be established. The top-level controlling object is a BPELProcess object.

The following example demonstrates how to import a BPEL file:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.bpel.BPELRegistrator;

String dbURL = "http://localhost:53307/CentraSite/CentraSite";
String bpelFile = "c:/temp/MyBPEL.bpel";
String user = ...;
String password = ...;

JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);

try
{
   BPELRegistrator br = new BPELRegistrator(bpelFile, jaxr);
   br.register();
   int warnings = br.getWarningCount();    // are there unreferenced services?
}
catch (Exception e)
{
   // handle error
}
finally
{
   jaxr.close();
}
```

The BPELRegistrator class provides the following optional setter methods. If you want to use them, they must be called before calling the register() method.

setProjectName

Specifies a project name, i.e. an internal RegistryPackage. The named project will receive the created top-level object (a BPELProcess) as a member.

Syntax

```
public void setProjectName(String projectName)
```

Parameters

String projectName
The desired name of the project.

setWarningIfPLTNotFound

Specifies how CentraSite should react if a service that has not yet been registered in CentraSite is encountered.

Syntax

```
public void setWarningIfPLTNotFound(boolean warningIfPLTNotFound)
```

Parameters

boolean warningIfPLTNotFound

true	Each time a service that has not yet been registered in CentraSite is encountered, a counter is incremented. Use the getWarningCount() method to get the current value of the counter.
false	If a service that has not yet been registered in CentraSite is encountered, an exception is thrown.

Usage Notes

The default value of the parameter is `true`.

Removing a Registered BPEL Object

You can remove a registered BPEL process object. The BPEL file is removed from the repository, and all associated registry objects are also removed. Like the import class, you must first establish a CentraSite connection with an instance of the JAXRAccessor class before calling the remove or removeProcess method. The BPELAdministrator class provides the following methods:

remove

Removes the BPEL objects of the process flow indicated in the specified BPEL file.

Syntax

```
public boolean remove (String bpelFile)
```

Parameters

String bpelFile
Specifies the BPEL file whose objects are to be removed. This could be the file in the CentraSite repository.

Return Codes

Value	Meaning
true	The BPEL objects were successfully removed.
false	The BPEL objects could not be removed.

removeProcess

Removes the BPEL objects of the process flow indicated by the BPELProcess name and its namespace.

Syntax

```
public boolean removeProcess(String bpelProcessName, String bpelNamespace
```

Parameters

String bpelProcessName
The process name of the BPEL objects to be removed.

String bpelNamespace
The namespace of the BPEL objects to be removed.

Return Codes

Value	Meaning
true	The BPEL objects were successfully removed.
false	The BPEL objects could not be removed.

Importing an XPDL File

The XPDL importer imports a process definition from an XPDL file. From the XPDL file, the importer produces a Process object and related components (e.g., Process Steps, Process Pools and Process Swimlanes). If the XPDL process references a Web service, the importer will add the Web service to the registry if it is not already present and associate it with the Process object.

The following example demonstrates how to import an XPDL file into CentraSite using the CentraSite Java API:

```
import com.centrasite.jaxr.JAXRAccessor;
import com.centrasite.jaxr.xpdl.ImportXPDL;

// Set URL for CentraSite Registry/Repository
String dbURL = "http://localhost:53307/CentraSite/CentraSite";

// Specify the location of the XPDL file.
String xpdlFile = "c:/temp/MyXPDL.xpdl";

// Specify the user account and password that the importer will use to
// log on to CentraSite
String user = "ctambrose";
String password = "jm6A1999";

// Build the connection to CentraSite
JAXRAccessor jaxr = new JAXRAccessor(dbURL, user, password);
```

```
try
{
// Instantiate XPDL importer object with specified XPDL file and
// connection info and then execute the import method.

   ImportXPDL xpdl = new ImportXPDL(xpdlFile, jaxr);
   xpdl.doImport();
}
catch (Exception e)
{
// Handle error
   ...
}
finally
{
// Close connection to CentraSite Registry/Repository
   jaxr.close();
}
```

The ImportXPDL class provides the following optional setter methods that you can use to specify certain properties in the Process object. If you want to use these setter methods, you must call them before you call the doImport() method.

setOrganization

Specifies the organization to which the Process object is to be added.

Syntax

```
public void setOrganization (Organization org)
```

Parameters

Organization org
The organization to which the importer will add the Process object. Note that this method takes an Organization object as input. You can obtain the Organization object for a specified organization using the BusinessQueryManager.getRegistryObject() method.

Usage Notes

Optional. If you do not set the organization parameter, the importer will add the Process object to the organization that the user specified in the JAXRAccessor belongs.

SetOriginalFilename

Specifies the filename to be assigned to the XPDL file in CentraSite's repository.

Syntax

```
public void setOriginalFilename (String originalFilename)
```

Parameters

String originalFilename
The filename that is to be given to the XPDL file in the CentraSite repository.

Note: A valid filename can consist of letters, numbers, and the underscore character. It must not contain spaces or other special characters.

Usage Notes

Optional.

setProductConcept

Specifies the category from the Product taxonomy by which the Process object is to be classified.

Syntax

```
public void setProductConcept(Concept product)
```

Parameters

Concept product
The Product category (concept) that is to be assigned to the Process object. This classification is generally used to identify the product from which the Process was published.

The following are some of the predefined categories in the Product taxonomy in CentraSite. (For other categories, examine the Product taxonomy on the instance of CentraSite to which you are importing the XPDL file.)

- CentraSite
- webMethods ApplinX
- webMethods EntireX
- webMethods Product Suite, with the subcategories that include:
  - webMethods BPM
  - webMethodsComposite Application Framework
  - webMethods Trading Networks

Usage Notes

Optional.

setVersion

Specifies the version identifier that is to be assigned to the Process object that the importer adds to the registry. (This methods specifies the user-defined *version identifier*. The Process object will also have a *version number*, which is automatically assigned and maintained by CentraSite.)

Syntax

```
public void setVersion(String version)
```

Parameters

String version
The version identifier to be assigned to the Process object.

Usage Notes

Optional.

Invoking an Importer from a Java Class

Each importer includes a main() method, which allows it to be called from a Windows batch file or from a UNIX shell script.

To invoke an importer from the command line, you must perform the following high-level steps:

1. Create a script file.
2. Execute the script file with the appropriate input parameters.

Creating a Script File to Invoke an Importer

The importers are Java classes whose main() method executes when you run the importer class from the command line. To ensure that the CLASSPATH and other environment variables are set properly when you run the importer class, you must create a script file.

Creating a Script File for Windows (a .bat File)

Create a script file that looks as follows if CentraSite is running under Windows.

```
@echo off
set JAVAEXE=fullPathToJava.exe
set REDIST=CentraSiteHomeDirectory\redist
set BASEDIR=%~dp0
cd /d %REDIST%

REM build CLASSPATH with all files from jar directory
set LOCAL_CLASSPATH=
for %%I in (".\*.jar") do call "CentraSiteHomeDirectory\bin\cfg\lcp.cmd" %%I

%JAVAEXE% -cp %LOCAL_CLASSPATH% importerClassName %*
cd /d %BASEDIR%
```

Where *importerClassName* is the name of the Importer class that you want to run. For a list of the importer class names, see "Importer Class Names" on page 190.

Example

The following is an example of a script file that calls the XML Schema importer:

```
@echo off
REM
REM Run XML Schema Importer
REM
set JAVAEXE=D:\software\java\jdk1.5.0_12\bin\java
set REDIST=C:\SoftwareAG\CentraSite\redist
set BASEDIR=%~dp0
cd /d %REDIST%

REM build CLASSPATH with all files from jar directory
set LOCAL_CLASSPATH=
for %%I in (".\*.jar") do call "C:\SoftwareAG\CentraSite\bin\cfg\lcp.cmd" %%I

%JAVAEXE% -cp %LOCAL_CLASSPATH% com.centrasite.jaxr.schema.SchemaImporter %*
cd /d %BASEDIR%
```

Creating a Script File for UNIX (C-Shell Script)

Create a script file that looks as follows if CentraSite is running under UNIX.

```
set javaexe="fullPathToJava.exe "
set redist="CentraSiteHomeDirectory /redist"
set mainjar="CentraSiteUtils.jar"
set delim='\:'
cd "$redist"
set cl=""
foreach j ( 'ls *.jar' )
  if ($cl != "") set cl=${cl}${delim}
  set cl=${cl}${j}
end
setenv CLASSPATH ${mainjar}${delim}${cl}
$javaexe importerClassName  $*
```

Where *importerClassName* is the name of the Importer class that you want to run. See "Importer Class Names" on page 190 for a list of the importer class names.

Example

The following is an example of a script file that calls the XML Schema importer:

```
#!/bin/csh
#
# Run XML Schema Importer
#
set javaexe="/mydir/softwareag/cjp/v16/bin/java"
set redist="/mydir/softwareag/CentraSite/redist"
set mainjar="CentraSiteUtils.jar"
set delim='\:'
# build CLASSPATH with all files from jar directory
cd "$redist"
set cl=""
foreach j ( 'ls *.jar' )
if ($cl != "") set cl=${cl}${delim}
set cl=${cl}${j} end
setenv CLASSPATH ${mainjar}${delim}${cl}
$javaexe com.centrasite.jaxr.schema.SchemaImporter $*
```

Importer Class Names

The following are the class names for the importers:

Note: Some importers have an *import class*, which you use to import the asset, and an *admin utility class*, which you can use to delete or replace the asset after it has been imported.

Importer or Admin Utility that You Want to Use	Class Name
Web Service (Importer)	com.centrasite.jaxr.webservice.WebServiceRegistrator
Web Service (Admin utility)	com.centrasite.jaxr.webservice.WebServiceAdministrator
XML Service (Importer)	com.centrasite.jaxr.xmlservice.XMLServiceManager
REST Service (Importer)	com.centrasite.jaxr.xmlservice.XMLServiceManager
XML Schema (Importer)	com.centrasite.jaxr.schema.SchemaImporter
XML Schema (Admin utility)	com.centrasite.jaxr.schema.SchemaAdministrator
BPEL Process Definition (Importer)	com.centrasite.jaxr.bpel.BPELRegistrator
BPEL Process Definition (Admin utility)	com.centrasite.jaxr.bpel.BPELAdministrator
XPDL File (Importer)	com.centrasite.jaxr.xpdl.ImportXPDL

Executing the Script File That Invokes an Importer

To invoke the importer, you must run the importer script file with the required set of input parameters. The input parameters that are required to run the script file will vary depending on the importer your script file invokes. (If you need information about creating the script file, see "Creating a Script File to Invoke an Importer" on page 188.)

Importing a Web Service

To import a Web service, run your importer script file with the following required input parameters:

```
yourScriptFile -w wsdlFile -o orgName
-user yourCSUserID -password yourPassword
```

Example

```
myScript -w d:\myDirectory\myWSDLFile.wsdl -o Customer Service
-user jcambrose -password j45Hk19a
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the Web Service importer:

Parameter	Description
`-w` *`wsdlFile`*	*Required*. The absolute or relative path to the WSDL file that you want to import. If relative, the path should be relative to the location from where the command is executed.
`-o` *`orgName`*	*Required*. The name of the organization into which you want to import the Web service. If the organization name contains a space, enclose the name in double-quotes.
`-user` *`yourCSUserID`*	*Required*. Your CentraSite user ID.
`-password` *`password`*	*Required*. The password for your CentraSite user account.
`-h` *`hostName`*	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-p` *`portNumber`*	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
`-dburl` *`url`*	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`.

Note: You can also obtain the complete list of input parameters for the Web Service importer by invoking the importer with no input parameters.

Invoking the Web Service Administrator from the Command Line

You can invoke the Web Service Administrator utility to delete a Web service that has been imported into CentraSite. To obtain the input parameters used by this utility, run the script file with no input parameters.

Importing an XML Schema

To import an XML Schema, run your importer script file with the following input parameters:

```
yourScriptFile -s xsdFile
-user yourCSUserID -password yourPassword
```

Example

```
myScript -s d:\myDirectory\myXSDFile.xsd -user jcambrose -password j45Hk19a
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the XML Schema importer:

Parameter	Description
`-s` *`xsdFile`*	*Required*. The absolute or relative path to the schema file that you want to import. If relative, the path should be relative to the location from where the command is executed.
`-user` *`userID`*	*Required*. Your CentraSite user ID.
`-password` *`password`*	*Required*. The password for your CentraSite user account.
`-h` *`hostName`*	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-p` *`portNumber`*	The port number on which the CentraSite registry/repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.

Parameter	Description
`-dburl url`	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/ CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and - p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
`-noover`	Prevents the importer from overwriting a schema if it is already present in the registry.

Note: You can also obtain the complete list of input parameters for the XML Schema importer by invoking the importer with no input parameters.

Invoking the XML Schema Administrator from the Command Line

You can invoke the XML Schema Administrator utility to delete an XML Schema that has been imported into CentraSite. To obtain the input parameters used by this utility, run the script file with no input parameters.

Importing an XML Service

To import an XML Service, run your importer script file with the following input parameters:

```
yourScriptFile -s xmlFileURI -n serviceName
-e endpointURL -m httpMethods -dbuser yourCSUserID
-dbpassword yourPassword
```

Example

```
myScript -s http://fs02hq/xml/myService.xsd -n myXMLService -e
http://appsvr02:53307/myService -m GET PUT -dbuser jcambrose -dbpassword j45Hk19a
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the XML Service importer:

Parameter	Description
`-s` *`xsdFile`*	*Required*. The absolute or relative path to the schema file from which the XML service is created. If relative, the path should be relative to the location from where the command is executed.
`-n` *`serviceName`*	*Required*. The name that is to be assigned to the service.
`-e` *`endpointURL`*	*Required*. The URL where the service is deployed. (The access point for the service.)
`-m` *`httpMethods`*	*Required*. The HTTP access methods that the service supports. Valid values are: GET, POST, PUT, DELETE. If the service supports multiple methods, list the methods separated by spaces (e.g., `-m GET PUT DELETE`).
`-dbuser` *`userID`*	*Required*. Your CentraSite user ID. (This user account must have permission to create assets in the organization to which the service will be added.)
`-dbpassword` *`password`*	*Required*. The password for your CentraSite user account.
`-dbhost` *`hostName`*	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-dbport` *`portNumber`*	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
`-suser` *`xsdFileUser`*	The user ID that is to be used if the XSD file (specified in the -s parameter) resides on a secure server and the importer is required to provide a user ID and password to access it.
`-spassword` *`xsdFilePassword`*	The password for the user account in `-suser`.

Parameter	Description
`-noover`	Do not overwrite existing imported XML Schemas that exist in the registry.
`-desc` *`description`*	The description that is to be assigned to the XML Service. If you omit this parameter, the service description will be empty.
`-ver` *`versionID`*	The version identifier that is to be assigned to the XML Service. (This is the user-defined identifier, not the system-assigned version number.) If you omit this parameter, the version identifier will be set to 1.0.
`-porg` *`providingOrg`*	The name of the organization that is to be designated as the providing organization for this service. If you omit this parameter, the providing organization will be set to the Default Organization.
`-sorg` *`submittingOrg`*	The name of the organization to which this service is to be added. If you omit this parameter, the service will be added to the Default Organization.

Note: You can also obtain the complete list of input parameters for the XML Service importer by invoking the importer with no input parameters.

Importing a REST Service

To import a REST Service, run your importer script file with the following input parameters:

```
yourScriptFile -s xmlFileURI -n serviceName
-e endpointURL -m httpMethods -dbuser yourCSUserID
-dbpassword yourPassword
```

Example

```
myScript -s http://fs02hq/xml/myService.xsd -n myXMLService -e
http://appsvr02:53307/myService -m GET PUT -dbuser jcambrose -dbpassword j45Hk19a
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the REST Service importer:

Parameter	Description
`-s` *`xsdFile`*	*Required*. The absolute or relative path to the schema file from which the REST service is created. If relative, the path should be relative to the location from where the command is executed.
`-n` *`serviceName`*	*Required*. The name that is to be assigned to the service.
`-e` *`endpointURL`*	*Required*. The URL where the service is deployed. (The access point for the service.)
`-m` *`httpMethods`*	*Required*. The HTTP access methods that the service supports. Valid values are: GET, POST, PUT, DELETE. If the service supports multiple methods, list the methods separated by spaces (e.g., `-m GET PUT DELETE`).
`-dbuser` *`userID`*	*Required*. Your CentraSite user ID. (This user account must have permission to create assets in the organization to which the service will be added.)
`-dbpassword` *`password`*	*Required*. The password for your CentraSite user account.
`-dbhost` *`hostName`*	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-dbport` *`portNumber`*	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
`-suser` *`xsdFileUser`*	The user ID that is to be used if the XSD file (specified in the -s parameter) resides on a secure server and the importer is required to provide a user ID and password to access it.
`-spassword` *`xsdFilePassword`*	The password for the user account in `-suser`.

Parameter	Description
`-noover`	Do not overwrite existing imported XML Schemas that exist in the registry.
`-desc` *`description`*	The description that is to be assigned to the REST Service. If you omit this parameter, the service description will be empty.
`-ver` *`versionID`*	The version identifier that is to be assigned to the REST Service. (This is the user-defined identifier, not the system-assigned version number.) If you omit this parameter, the version identifier will be set to 1.0.
`-porg` *`providingOrg`*	The name of the organization that is to be designated as the providing organization for this service. If you omit this parameter, the providing organization will be set to the Default Organization.
`-sorg` *`submittingOrg`*	The name of the organization to which this service is to be added. If you omit this parameter, the service will be added to the Default Organization.

Note: You can also obtain the complete list of input parameters for the REST Service importer by invoking the importer with no input parameters.

Importing a BPEL Process

To import a BPEL process, call your importer script file with the following input parameters:

```
yourScriptFile -file bpelFile -user yourCSUserID
-password yourPassword
```

Example

```
myScript -file d:\myDirectory\myBPELFile.bpel -user jcambrose -password j45Hk19a
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the BPEL importer:

Parameter	Description
`-file` *`bpelFile`*	*Required*. The absolute or relative path to the BPEL process file that you want to import. If relative, the path should be relative to the location from where the command is executed.
`-user` *`userID`*	*Required*. Your CentraSite user ID.
`-password` *`password`*	*Required*. The password for your CentraSite user account.
`-h` *`hostName`*	*Optional.* The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-p` *`portNumber`*	The port number on which the CentraSite registry/repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
`-dburl` *`url`*	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
`-nowarning`	Suppresses the warning message that the importer issues if the BPEL process references a BPEL Partner Link Type that refers to a Web service that is not present in the CentraSite registry.

Note: You can also obtain the complete list of input parameters for the BPEL importer by invoking the importer with no input parameters.

Invoking the BPEL Administrator from the Command Line

You can invoke the BPEL Administrator utility to delete or display a BPEL process that has been imported into CentraSite. To obtain the input parameters used by this utility, invoke the BPEL administrator with no input parameters.

Importing an XPDL File

To import an XPDL file, call the importer script file from the command line as shown below.

```
yourScriptFile -file xpdlFile -user yourCSUserID
-passwordyourPassword
```

Example

```
myScript -file d:\myDirectory\myXPDLFile.xpdl -user jcambrose -password j45Hk19a
```

Input Parameters

The following table describes input parameters that you can use with the XPDL importer:

Parameter	Description
`-file` *`xpdlFile`*	*Required*. The absolute or relative path to the XPDL file that you want to import. If relative, the path should be relative to the location from where the command is executed.
`-user` *`userID`*	*Required*. Your CentraSite user ID.
`-password` *`password`*	*Required*. The password for your CentraSite user account.
`-h` *`hostName`*	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
`-p` *`portNumber`*	The port number on which the CentraSite registry/repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.

Note: You can also obtain the complete list of input parameters for the XPDL importer by calling the script file with no input parameters.

Invoking an Importer From the Command Line

Each importer includes a command line tool, which allows it to be executed from the command prompt (.cmd for Windows) or a terminal (.sh for any UNIX flavor). The command line tool is located in <*CentraSiteInstallDir* >/utilities.

In both cases, the command is followed by a list of appropriate input parameters and options. The parameters are key-value pairs, where the key starts with a hyphen. An option is a switch that modifies the behavior of the command and is preceded by a hyphen. Using the -usage option lists all available input parameters and options with a short description.

If you start the command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

Importing a Web Service

To import a Web service in CentraSite registry, use a command `ImportWSDL` of the following format:

On Windows

```
ImportWSDL.cmd -user <USERNAME> -password <PASSWORD> [-dburl <CENTRASITE-URL> |
[[-h <HOSTNAME>] [-p <PORT>]]] -o <ORGANIZATION> -w <WSDL-FILE-PATH>
```

On UNIX

```
ImportWSDL.sh -user <USERNAME> -password <PASSWORD> [-dburl <CENTRASITE-URL> |
[[-h <HOSTNAME>] [-p <PORT>]]] -o <ORGANIZATION> -w <WSDL-FILE-PATH>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `ImportWSDL` utility:

Parameter	Description
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
CENTRASITE-URL	The fully qualified URL for the CentraSite registry/ repository.

Parameter	Description
	If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
HOSTNAME	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
PORT	The port number on which the CentraSite registry/repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
ORGANIZATION	*Required*. The name of the organization into which you want to import the Web service. If the organization name contains a space, enclose the name in double-quotes.
WSDL-FILE-PATH	*Required*. The absolute or relative path to the WSDL file that you want to import. If relative, the path should be relative to the location from where the command is executed.

Importing an XML Schema

To import an XML Schema in CentraSite registry, use a command `ImportSchema` of the following format:

On Windows

```
ImportSchema.cmd -user <USERNAME> -password <PASSWORD> [-dburl <CENTRASITE-URL> |
[[-h <HOSTNAME>] [-p <PORT>]]] -s <XSD-FILE> [-noover]
```

On UNIX

```
ImportSchema.sh -user <USERNAME> -password <PASSWORD> [-dburl <CENTRASITE-URL> |
[[-h <HOSTNAME>] [-p <PORT>]]] -s <XSD-FILE> [-noover]
```

Input Parameters and Options

The following tables describe the complete set of input parameters and options that you can use with the `ImportSchema` utility:

Parameter	Description
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
CENTRASITE-URL	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
HOSTNAME	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
PORT	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
XSD-FILE	*Required*. The absolute or relative path to the schema file that you want to import. If relative, the path should be relative to the location from where the command is executed.

Option	Description
`-noover`	Prevents the importer from overwriting an XML schema if it is already present in the registry.

Importing a BPEL Process

To import a BPEL Process in CentraSite registry, use a command `ImportBPEL` of the following format:

On Windows

```
ImportBPEL.cmd -user <USERNAME> -password <PASSWORD> [[-dburl CENTRASITE-URL]|
[[-h <HOSTNAME>] [-p <PORT>]]] -file <BPEL-FILE> [-nowarning]
```

On UNIX

```
ImportBPEL.sh -user <USERNAME> -password <PASSWORD> [[-dburl CENTRASITE-URL]|
[[-h <HOSTNAME>] [-p <PORT>]]] -file <BPEL-FILE> [-nowarning]
```

Input Parameters and Options

The following tables describe the complete set of input parameters and options that you can use with the `ImportBPEL` utility:

Parameter	Description
BPEL-FILE	*Required*. The absolute or relative path to the BPEL process file that you want to import. If relative, the path should be relative to the location from where the command is executed.
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
HOSTNAME	*Optional.* The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
PORT	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests.

Parameter	Description
	If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.
CENTRASITE-URL	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)

Option	Description
`-nowarning`	Suppresses the warning message that the importer issues if the BPEL process references a BPEL Partner Link Type that refers to a Web service that is not present in the CentraSite registry.

Importing an XPDL File

To import an XPDL file in CentraSite registry, use a command `ImportXPDL` of the following format:

On Windows

```
ImportXPDL.cmd -user <USERNAME> -password <PASSWORD>
[[-h <HOSTNAME>] [-p <PORT>]] -file <XPDL-FILE>
```

On UNIX

```
ImportXPDL.sh -user <USERNAME> -password <PASSWORD>
[[-h <HOSTNAME>] [-p <PORT>]] -file <XPDL-FILE>
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `ImportXPDL` utility:

Parameter	Description
XPDL-FILE	*Required*. The absolute or relative path to the XPDL file that you want to import. If relative, the path should be relative to the location from where the command is executed.
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
HOSTNAME	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
PORT	The port number on which the CentraSite registry/repository is configured to listen for incoming requests. If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53307`.

Importing an Archive

To import an archive in CentraSite registry, use a command `ImportArchive` of the following format:

On Windows

```
ImportArchive.cmd [-setreplace] [-importorg <ORGANIZATION-KEY>] [-keepowner]
[-keeporganization] <CENTRASITE_URL> <ARCHIVE_FILENAME> <USERNAME> <PASSWORD>
```

On UNIX

```
ImportArchive.sh [-setreplace] [-importorg <ORGANIZATION-KEY>] [-keepowner]
[-keeporganization] <CENTRASITE_URL> <ARCHIVE_FILENAME> <USERNAME> <PASSWORD>
```

Input Parameters and Options

The following table describes the complete set of input parameters and options that you can use with the `ImportArchive` utility:

Parameter	Description
ARCHIVE_FILENAME	*Required*. The absolute or relative path to the export archive file. If relative, the path should be relative to the location from where the command is executed. The archive file can contain an organization with its assets or can contain a set of objects that were exported from one or more organizations.
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
CENTRASITE-URL	*Required*. The fully qualified URL for the CentraSite registry/repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53307/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
ORGANIZATION-KEY	By default, objects are imported into the current user's organization. Use this optional parameter if you want to specify a different organization into which the objects should be imported. The keyword is followed by the organization's GUID, prefixed by "uddi:", for example: "uddi:207ff1cc-25c5-544c-415c-5d98ea91060c". **Note:** This parameter has no effect if the –`keeporganization` option is set.

Option	Description
-setreplace	When an object is to be imported, the timestamp of the object in the import archive file is compared with the timestamp of the corresponding object with the same GUID in the registry, if the latter exists. If the timestamps are equal, the object is not imported. If the

Option	Description
	archive timestamp is older than the registry timestamp, the object is only imported if this optional parameter is present; otherwise it is ignored.
-keepowner	This option will maintain the original owner of the importing objects. This requires the original owner to exist with the same UDDI key on the target machine. If the parameter is not set, the importing user will be owner of the objects.
-keeporganization	This option will maintain the original organization of the importing objects. This requires the original organization to exist with the same UDDI key on the target machine. **Note:** If an organization is imported with its assets, this option is automatically activated.

Invoking an Importer Using the SOAP API

CentraSite provides a Web service for each of the predefined importers. Descriptions of these services are available here:

`http://server:port/wsstack/services/listServices`

where *server* is the machine on which the Software AG Runtime is running and *port* is the port on which Tomcat is listening (port 53307 if CentraSite is configured to use the default Software AG Runtime port number).

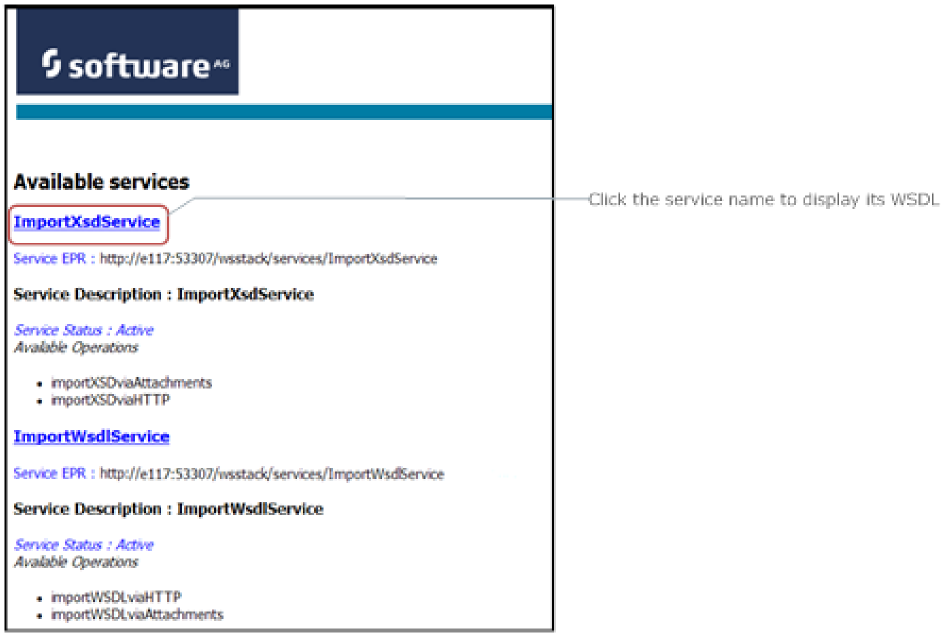
Example

```
http://myServer:53307/wsstack/services/listServices
```

We recommend using MTOM, when using the CentraSite web services with attachments of 1 megabyte or more.

Viewing the WSDL for the Importers

To view the WSDL for an importer service, click the service name on the listServices page.



Available services

ImportXsdService

Service EPR : http://e117:53307/wsstack/services/ImportXsdService

Service Description : ImportXsdService

Service Status : Active
Available Operations

- importXSDviaAttachments
- importXSDviaHTTP

ImportWsdlService

Service EPR : http://e117:53307/wsstack/services/ImportWsdlService

Service Description : ImportWsdlService

Service Status : Active
Available Operations

- importWSDLviaHTTP
- importWSDLviaAttachments

Writing Your Own Importer

You can write your own plug-in for CentraSite Control to incorporate your own importer. The prepared plug-in is a collection of files in a specific directory structure. After implementing the plug-in, the files must be copied into the CentraSite Controlwebapps folder under:

`<RuntimeWebAppsDir>/PluggableUI/<MyPluginFolder>`

The location of the <*RuntimeWebAppsDir*> folder is described in "Basic Operations" on page 19.

The folder <*MyPluginFolder*> must contain the following files and folders:

Name of File or Folder	Description
plugin.xml	Top plug-in description file
*.html	Files generated for the GUI
*_SWT.xml	Files generated for the GUI
images	Directory for icons (GIF files)
lib	Directory for JAR files provided by the plug-in

Name of File or Folder	Description
accesspath	Directory created by HTML generation

The ImportMyFile framework illustrates how an import plug-in may be set up. The example extends the import selection list and presents a screen that prompts for the file to be imported. After confirming the file, the appropriate adapter classes are called.

You may use this example as a guideline, adapting it and renaming it to suit your individual requirements. The templates indicate where customization is required.

The Build Environment

This topic explains the build environment for generating the HTML files that are used for the GUI and for compiling the necessary Java source files. It assumes the use of Ant, the Java-based build tool.

The following file system structure under the plug-in directory is assumed:

Name of File or Folder	Description
src	The directory that holds the Java source files
xml	The directory that holds the XML file that specifies your import window
plugin.xml	Top plugin description file that specifies the extension point and the command class
build.xml	The Ant input file for building the destination files

The Ant file shown below, named build.xml, can be used to establish an import plug-in. Look for the properties with the following names:

plugin.name

plugin.provider

tomcat.dir

tomcat.ver.dir

and modify them as required to suit your installation.

This is build.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="./ant2html.xsl"?>
<!--
Build an Import plugin
-->
```

```
<project name="Import Plugin Example" default="all" basedir=".">
  <description> Build file for an Importer plugin </description>

  <!-- environment -->
  <property environment="env"/>

  <property name="plugin.name" value="ImportMyFile" />
  <property name="plugin.provider" value="Software AG" />

  <!-- tomcat home directory -->
  <!-- <property name="tomcat.dir"
            value="tomcat directory of installation" />   -->
  <property name="tomcat.dir" value="C:/SoftwareAG/profiles/CTP" />
  <property name="tomcat.ver.dir" value="${tomcat.dir}/workspace"/>

<!-- point to the root of the pluggableUI to get the cis environment -->
<property name="pluggable.ui.root"
         value="${tomcat.ver.dir}/webapps/PluggableUI"/>

<!-- the directory containing source code -->
<property name="plugin.dir" value="../${plugin.name}" />
<property name="src.dir"    value="${plugin.dir}/src" />
<property name="xml.dir"    value="${plugin.dir}/xml" />
<property name="classes"    value="${plugin.dir}/classes" />
<property name="lib"        value="${plugin.dir}/lib" />

<!-- classpath -->
<path id="plugin.class.path">
  <fileset dir="${pluggable.ui.root}/CentraSiteControl/lib">
    <include name="*.jar"/>
  </fileset>
  <fileset dir="${pluggable.ui.root}/WEB-INF/lib">
    <include name="*.jar"/>
  </fileset>
</path>

<!-- default target, build all -->
<target name="all" description="all" depends="jar, zip"/>

<!-- establish jar file of plugin -->
<target name="jar" depends="compile" description="jar">
  <mkdir dir="${lib}" />
  <jar destfile="${lib}/${plugin.name}.jar">
    <fileset dir="${classes}"/>
    <fileset dir="${src.dir}" includes="**/*.properties"/>
    <manifest>
      <section name="com/centrasite/control">
        <attribute name="Implementation-Title" value="${plugin.name}"/>
        <attribute name="Implementation-Version" value="1.0.0.0"/>
        <attribute name="Implementation-Vendor"
                   value="${plugin.provider}"/>
      </section>
    </manifest>
  </jar>
</target>

<!-- compile java sources -->
<target name="compile" description="compile" depends="">
  <mkdir dir="${plugin.dir}/accesspath" />
  <mkdir dir="${classes}" />
  <javac srcdir="${src.dir}" destdir="${classes}" debug="on"
         classpathref="plugin.class.path" />
```

```
</target>

<!-- Create plugin archive -->
<target name="zip" description="package the plugin" depends="">
  <zip destfile="${plugin.name}.zip" basedir="${plugin.dir}/.."
                                     includes="${plugin.name}/lib/**,
                 ${plugin.name}/accesspath/**, ${plugin.name}/plugin.xml,
                                               ${plugin.name}/*_SWT.xml,
                 ${plugin.name}/xml/** ${plugin.name}/*.html" />
</target>

<!-- Install plugin zip to PluggableUI -->
<target name="install" description="Install plugin zip to PluggableUI">
    <java classname=
          "com.softwareag.cis.plugin.ext.plugins.command.InstallPlugInCommand"
          fork="true">
          <arg value="-t" />
          <arg value="${pluggable.ui.root}" />
          <arg value="-z" />
          <arg value="${plugin.name}.zip" />
          <classpath>
              <fileset dir="${pluggable.ui.root}/WEB-INF/lib">
                  <include name="*.jar"/>
             </fileset>
          </classpath>
    </java>
</target>

<!-- Uninstall plugin zip from PluggableUI -->
<target name="uninstall" description="Uninstall plugin zip from PluggableUI">
  <delete dir="${pluggable.ui.root}/${plugin.name}" />
</target>

<!-- Cleanup objects -->
<target name="clean" description="clean classes lib and generated files">
  <delete dir="${classes}" />
  <delete dir="${plugin.dir}/accesspath" />
  <delete dir="${lib}" />
  <delete file="${plugin.name}.zip" />
</target>

</project>
```

The classpath for the build step must comprise all JAR files used by the UI. Add these JAR files to the build path of your java project also.

As mentioned above, in order to present a user-defined screen when the plug-in's **import** button is clicked, an XML file that describes the GUI must be located in the subdirectory xml. The example XML file (xml/ImportMyFile.xml) simply prompts for a filename:

```
<?xml version="1.0" encoding="UTF-8"?>
<page model="com.importer.myfile.control.ImportMyFileAdapter">
  <pagebody>
      <rowarea withleftborder="false" withtopborder="false"
               withrightborder="false" withbottomborder="false"
               withtoppadding="false"
               paddingleft="10" paddingright="10">
          <vdist height="30"></vdist>
          <itr>
              <label name="File:" width="100" asplaintext="true"></label>
              <fileupload2 width="100%" cfileprop="fileClientUrl"
                           fileprop="fileServerUrl"
                           method="fileLoaded"></fileupload2>
          </itr>
          <vdist height="30"></vdist>
      </rowarea>
  </pagebody>
  <statusbar></statusbar>
</page>
```

The Plug-In Environment

The master file of the plug-in is plugin.xml:

```
<plugin id="com.importer.myfile" order="101">
  <requiredPlugin id="com.centrasite.control" />
  <requiredPlugin id="com.softwareag.cis.plugin" />

  <!-- Import extension (point to command execution class) -->
  <extension point="com.centrasite.control.import"
            id="importMyFileCommand"
            class="com.importer.myfile.control.ImportMyFileCommand">
  </extension>
</plugin>
```

This file specifies the command class, which is used to select the user's import function. It must be derived from com.centrasite.control.extpt.AbstractImport.

Note: This topic does not explain all the details of the Java source file; its purpose is to indicate the code that must be modified to suit your environment.

Here is src/com/importer/myfile/control/ImportMyFileCommand.java:

```
package com.importer.myfile.control;

import com.centrasite.control.extpt.AbstractImport;
import com.centrasite.control.ActionContext;
public class ImportMyFileCommand extends AbstractImport
{
  static final String
     IMPORT_NAME     = "Import MyFile";  // Appears in the import list
  static final String
     HTML_PAGE       = "/ImportMyFile/ImportMyFile.html";
  static final String
     MY_IMAGE        = "../ImportMyFile/images/importMyFile.gif";
  static final String
     CALLING_ADAPTER = "com.importer.myfile.control.ImportMyFileAdapter";
                         // point to the adapter class

  public ImportMyFileCommand() {
  }

  public String getName() {
    return IMPORT_NAME;
  }

  public String getImageURL() {
    return MY_IMAGE;
  }

  public String getLayout() {
    return HTML_PAGE;
  }
  public String getPageDescription() {
    return "XPDL v.1 Importer";
  }

  public void execute(ActionContext actionContext) {
    actionContext.showPage(HTML_PAGE, getName(), CALLING_ADAPTER);
  }
}
```

This class defines the paths of the image file for your private icon, the HTML file used and the class of the import adapter.

Here is the frame of an import adapter (this is src/com/importer/myfile/control/ImportMyFileAdapter.java):

```
package com.importer.myfile.control;

import java.util.Collection;
import javax.xml.registry.JAXRException;
import com.centrasite.control.AbstractBrowseCommand;
import com.centrasite.control.ActionContext;
import com.centrasite.control.Connector;
import com.centrasite.control.adapters.BaseAdapter;
import com.centrasite.control.adapters.util.ImportAdapter;
import com.centrasite.control.discovery.PromptYesNoHandler;
import com.centrasite.control.logged.action.LoggedExecutor;
import com.centrasite.control.logged.action.LoggedSchemaImport;
import com.centrasite.control.interfaces.Initializable;
import com.centrasite.jaxr.JAXRAccessor;
import com.softwareag.cis.plugin.interfaces.RunnableDeferred;

/**
   *  Import adapter
   */
public class ImportMyFileAdapter extends BaseAdapter
             implements Initializable,ImportAdapter
{
   private static final String TITLE = "Import MyFile";

   private String fileTmpUrl;
   private String fileAtServer;
   private String fileAtClient;

   public ImportMyFileAdapter() {
       fileAtServer = fileAtClient = fileTmpUrl = null;
   }
   public void initialize(Collection<Object> objs){}

   public void setOrganization(String org){}

   public boolean execute() {
       callFinish();
       return true;
   }

   public String getFileClientUrl() {
       return fileAtClient;
   }

   public void setFileClientUrl(String value) {
       fileAtClient = value;
   }
```

```
public String getFileServerUrl() {
    return fileTmpUrl;
}

public void setFileServerUrl(String value) {
    fileTmpUrl = value;
}

public void fileLoaded() {
  fileAtServer = fileTmpUrl;
}

public void callCancel() {
  super.endProcess();
}

private static boolean isWhiteSpace(String s)
{
    if (s == null || s.length() == 0) return true;
    for (int i=0 ; i < s.length() ; ++i) {
        if (!Character.isWhitespace(s.charAt(i))) return false;
    }
    return true;
}

/**
 * Called if "OK" button has been pressed
 */
public void callFinish() {
    if (isWhiteSpace(fileAtServer))
    {
        outputMessage(MT_ERROR, "no file entered");
    }
    else
    {
       ImportMyFile ie = new ImportMyFile(getConnector(),
                              fileAtServer, fileAtClient);
       ie.doImport();
    }
}

/**
 * Class for importing specific files
 */
private class ImportMyFile extends AbstractBrowseCommand
              implements RunnableDeferred, PromptYesNoHandler
{
    private Connector connector;
    private JAXRAccessor jaxr;

    public ImportMyFile(Connector connector, String fileAtServer,
                        String fileAtClient){
       // fileAtClient is the filename you can access
       this.connector = connector;
       this.jaxr = null;
    }

    public void doImport() {
       JAXRAccessor jaxr = null;
       try {
          LoggedSchemaImport lsi =
              new LoggedSchemaImport(getActionContext());
          jaxr = getJAXR();
```

```
            // write here your import code to registry and repository
            //setEventCallback( lsi.getEventCallback() );
                 // for event logging

            // your import code can use the methods
            // accept(javax.xml.registry.infomodel.RegistryObject ro)
            // warning(javax.xml.registry.infomodel.RegistryObject ro)
            // reject(javax.xml.registry.infomodel.RegistryObject ro)
            // of the LoggedEventCallback class,
            // to log the status of your import

            new LoggedExecutor(lsi).execute();
         }
         catch (Exception e) {
            throw new RuntimeException(e);
         }
         finally {
            if (jaxr != null)
               jaxr.close();
         }
      }

      private JAXRAccessor getJAXR() throws JAXRException {
          if (jaxr == null)  {
              jaxr = new JAXRAccessor(connector.getRegistryUrl(),
                         connector.getUserName(),
                         connector.getPassword());
          }
          return jaxr;
      }
      public void run() throws Exception {
          doImport();
      }

      public void handleYes(ActionContext actionContext) {
          doImport();
      }

      public void handleNo(ActionContext actionContext) {}

      public void executeCommand(ActionContext actionContext,
                              String clientPath) {}

      public void executeCommand(ActionContext actionContext,
           String clientPath, String serverPath) {
           actionContext.executeDeferred(this);
      }

      public int getCategory() {
          return CATEGORY_MISC;
      }

      public String getImageURL() {
          return null;
      }

      public String getTitle() {
          return TITLE;
      }

      public String getName() {
          return "";
      }
```

```
        public String getLabel() {
            return "";
        }
    }
}
```

Assuming that you have set up all the Java files correctly in the directory src, you should be able to build with the command:

```
ant -f build.xml jar all
```

This creates the plug-in-specific JAR file in the subdirectory lib and archives the necessary plugin files into the file ImportMyFile.zip.

Propagating the Plug-In

Having generated the plugin files, they must be propagated into the directory PluggableUI of the installed CentraSite Control. Thus, for example, ImportMyFile (shown in "The Plug-In Environment" on page 212) should have the following directory/file structure:

```
.../PluggableUI/ImportMyFile /
  accesspath /
      ImportMyFile.access
  images /
      ImportMyFile.gif
  lib /
      ImportMyFile.jar
  ImportMyFile.html
  ImportMyFile_JLIBS.html
  ImportMyFile_SWT.xml
  plugin.xml
```

You can do this by executing the following command, which installs ImportMyFile.zip into the PluggableUI folder:

```
ant -f build.xml install
```

Generating all additionally needed files

You only need this step if you are not able to install the necessary files directly in the <*RuntimeWebAppsDir*> /PluggableUI/ direction. In this case, you have to build the structure shown above by yourself. You can automatically generate the .html files as well as the _SWT.xml and the .accesspath file using the Software AG Application Designer.

First edit the file stored in the following directory: <*RuntimeWebAppsDir*> /PluggableUI/ cis/config/cisconfig.xml.

Insert the statement: `plugindevelopment="true"` in the following part:

```
designtimeclassloader=
      "com.softwareag.cis.plugin.registry.loader.AdapterPluginClassLoader"
   enableadapterpreload="true"
   framebuffersize="3"
   plugindevelopment="true"
   loglevel=""
   logtoscreen="false"
   maxitemsinfieldcombo="100"
```

After that, restart the Software AG Runtime. Now you can start the Application Designer with the following URL: `http://localhost:53307/PluggableUI/HTMLBasedGUI/workplace/ide.html`.

Navigate to "Tools & Documentation > Layout Manager" and choose the "ImportMyFile" Plugin as Application Project. It appears in the Layout Definitions List. Click on it and generate all additionally needed files using the button "Operations on multiple Items > (Re)Generate HTML pages".

Activating the Plug-In

To activate the plug-in

1. Restart the Software AG Runtime.
2. Start CentraSite Control.
3. Choose the Import function. You will see the name of your plug-in in the **Import as** selection list.

   Select the name of your plug-in and click Next.
4. Enter the name of the file to be imported, then click Finish.

   Enter the name of the file to be imported, then click Finish.

Deactivating the Plug-In

To deactivate the plug-in

1. Run the command:

   ```
   ant -f build.xml uninstall
   ```
2. Restart the Software AG Runtime.
3. Start CentraSite Control.

6 Object Type Management

What Is a Type?

A *type* (also called an object type) is analogous to a class in object-oriented programming and describes a kind of object that the registry can store. Objects abstract the real-world entities and each object belongs to a particular type that defines its characteristics and behavior.

CentraSite includes many predefined types. Any custom type that you add to CentraSite is treated as an *asset type* (that is, instances of that type are treated as *assets*).

Types are system-wide objects, meaning that they apply to all organizations. Consequently, all organizations within a particular instance of CentraSite use (or have access to) the same global set of types.

A type is made up of *attributes*. Attributes hold data about an object. When a type defines an asset (that is, if it is an *asset type*), the attributes that make up the type are assigned to *profiles*. Profiles determine how the type's attributes are grouped and presented when an instance of that type is displayed in CentraSite Control, CentraSite Business UI or the CentraSite plug-in for Eclipse.

Asset types also include an additional set of properties called *advanced settings*. These properties determine how CentraSite manages instances of that type. Among other things, the advanced settings for an asset type determine whether assets of the type are visible in the catalog browser, whether they can be used with reports and/or policies, and whether they can be versioned.

Classifications of a Type

Broadly speaking, CentraSite classifies a type by the following categories: base type, sub type, and virtual type.

What Is a Base Type?

A base type is the core object of Asset Management in CentraSite. A base type, or sometimes referred to as a parent type, in an object-oriented architecture, is an object type from which other types are derived. It facilitates the creation of other types that can reuse the properties implicitly inherited from the base type. In addition, a user can extend base type functionality by adding or overriding properties relevant to the derived (sub) type.

For example, the Service object type is the base (parent) type for sub (derived) types such as REST Service, XML Service, Virtual Service, Virtual REST Service, and Virtual XML Service.

Properties of a Base Type

Base type is the highest type and does not inherit properties from any other type. Other types can inherit from a base type. They are called sub types. Base types have properties that differ from sub types.

The following properties are available for a base type:

- Large and small icons
- Visible in asset browse
- Enable reports
- Policies can be applied
- Require consumer registration
- Enable versioning
- Top level type
- Enable lifecycle management
- Visible in search
- Inherit base type profiles
- Inherit base type policies
- Inherit base type LCMs
- Exclude sub types from Business UI search

What Is a Sub Type?

A sub type is any type that inherits properties from any other sub type or base type. A sub type derived from a base type has the same set of attributes as its base type, but has its own set of profiles and properties and adds its own behavior. For example, Service is a base type from which REST Service and the XML Service are derived. REST Service and XML Service have basically the same set of metadata as the Service type, but each represents its own properties (for example, policies and LCMs) than the Service type.

A sub type can also have its own lifecycle model and policies.

Properties of a Sub Type

A sub type inherits properties from another sub type or base type. In addition, sub types have properties that differ from other types.

The following properties are available for a sub type:

- Large and small icons
- Visible in asset browse

- Enable reports
- Policies can be applied
- Require consumer registration
- Enable versioning
- Top level type
- Enable lifecycle management
- Visible in search
- Inherit base type profiles
- Inherit base type policies
- Inherit base type LCMs
- Clone base type profiles

What Is a Virtual Type?

A virtual type is a subset of its sub type. It inherits properties from its sub type, but has its own set of profiles and properties and adds its own behavior. A virtual type can also have its own lifecycle model and policies. In addition, you can configure a virtual type to inherit properties from its base type Service.

Virtual types do not have a separate storage structure or a schema, so the instances of a virtual type are stored as regular objects. For example, a Virtual Service inherits the Service and adds its own behavior, yet it is stored as a Service Object.

Technically speaking, a virtual type is the consumer-facing proxy for a sub type. For example, the Virtual Service type, Virtual XML Service type, and the Virtual REST Service type are the proxy types of the Service, XML Service, and REST Service types, respectively.

Several predefined virtual types are installed with CentraSite. For more information, see "Predefined Asset Types Installed with CentraSite" on page 263.

Properties of a Virtual Type

A virtual type inherits properties from its sub type and base type. In addition, virtual types have properties that differ from other types.

The following properties are available for a virtual type:

- Large and small icons
- Visible in asset browse
- Enable reports
- Policies can be applied

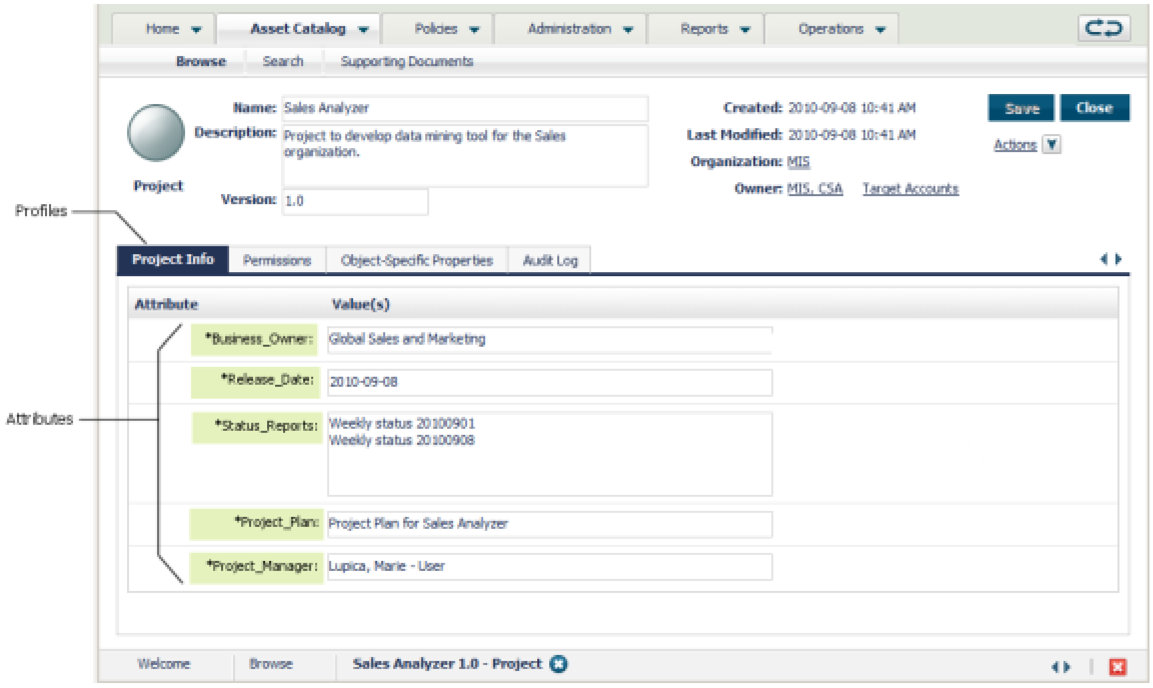
- Require consumer registration
- Enable versioning
- Top level type
- Enable lifecycle management
- Visible in search
- Inherit base type profiles
- Inherit base type policies
- Inherit base type LCMs
- Clone base type profiles

Basic Components of a Type

A type consists of two basic components:

- *Attributes* that represent an individual characteristic, property or piece of information about an asset.
- *Profiles* that represent a logical collection of attributes.

When displayed in the CentraSite Control graphical user interface



Name: Sales Analyzer
Description: Project to develop data mining tool for the Sales organization.
Version: 1.0

Created: 2010-09-08 10:41 AM
Last Modified: 2010-09-08 10:41 AM
Organization: MIS
Owner: MIS, CSA   Target Accounts

Attribute	Value(s)
*Business_Owner:	Global Sales and Marketing
*Release_Date:	2010-09-08
*Status_Reports:	Weekly status 20100901 Weekly status 20100908
*Project_Plan:	Project Plan for Sales Analyzer
*Project_Manager:	Lupica, Marie - User

Attributes

A type is made up of attributes. An attribute represents an individual characteristic, property, or piece of information about an asset. For example, the asset type for a Service includes attributes that identify the name of the service, provide the service's endpoints, identify the owner of the service, and supply links to programming documentation.

All types that represent assets include the following basic attributes:

Attribute	Description
Name	The name under which the asset is cataloged.
Description	A descriptive comment that provides additional information about an asset.
Key	The Universally Unique Identifier (UUID) that is assigned to the asset and uniquely identifies it within the registry. CentraSite automatically assigns a UUID to an asset when the asset is added to the registry.
Version	The user-assigned version identifier for an asset. The user-assigned identifier can include any sequence of characters. It is not required to be numeric.
System Version	The system-assigned version number that CentraSite maintains for its own internal use. CentraSite automatically assigns this identifier to an object when a version of the object is created. The system-assigned identifier is always numeric and always has the format: *MajorVersion .Revision* Where: ■ *MajorVersion* is an integer that represents the asset's version number. This value is incremented by one when a new version of the asset is generated (for example, 1.0, 2.0, 3.0). ■ *Revision* is an integer that represents an update to a particular version of an asset. When the revisioning feature is enabled for CentraSite, the *Revision* number is incremented each time a change is made to the object (for example, 1.0, 1.1, 1.2). An asset's **System Version** attribute cannot be deleted or modified by a user.

Attribute	Description
Created	The date on which the asset was added to the catalog. CentraSite automatically sets this attribute when a user adds the asset to the catalog. Once it is set, it cannot be modified.
Last Modified	The date on which the catalog entry for the asset was last updated. CentraSite automatically updates this attribute when a user modifies any of the asset's attributes.
Submitting Organization	The organization to which the asset belongs.
Owner	The user who currently owns the asset. CentraSite automatically sets this attribute when a user adds the asset to the catalog.
Lifecycle State	The asset's current lifecycle state. If a lifecycle model has been associated with an asset type, CentraSite updates this attribute as the asset passes through its lifecycle.

An asset can also have any number of additional attributes that are specific to the asset's type. For example, an asset might include attributes that do the following:

- Provide contact information for technical support (for example, phone numbers and email addresses)
- Classify the asset according to one or more taxonomies
- Describe an asset's relationship to other assets or registry objects
- Specify details regarding system requirements and technical specifications
- Provide links to additional information such as program documentation, sample code, or usage notes

Attribute Data Types

When you add an attribute to a type, you specify the attribute's *data type*. The data type determines what kind of information the attribute can hold. After you add an attribute to a type, the attribute's data type cannot be changed.

The following table lists the data types that you can assign to an attribute. Most types can be configured to hold a single value or multiple values (that is, an array of values).

Data Type	Description
Boolean	Holds a `true` or `false` value.

Data Type	Description
	Note: When a Boolean value is displayed in the CentraSite user interface, its value is generally displayed as "Yes" (if the attribute's value is true) or "No" (if the attribute's value is false).
Classification	Holds references to one or more categories in a specified taxonomy. You use this type of attribute to classify assets according to a specified taxonomy.
Computed Attribute	Holds a value that is supplied by a user-defined Java plug-in. After you have defined a computed attribute, you can use it in CentraSite Control in the same way as any other attribute. You can, for example, assign the attribute to a profile or reorder the attribute position within a profile.
Date/Time	Holds a timestamp that represents a specific date and/or time.
Duration	Holds a value that represents a period of time as expressed in Years, Months, Days, Hours, Minutes, and Seconds.
Email	Holds an email address. This data type only accepts values in the format: *anyString @anyString* **Note:** When a user enters a value for an Email attribute, CentraSite verifies that the value conforms to the format above, but it does not attempt to validate the address itself.
File	Holds references to one or more documents that reside in CentraSite's supporting document library or at a specified URL. You can use this type of attribute to attach documents such as programming guides, sample code, and other types of files to an asset.
IP Address	Holds a numeric IP address in the v4 or v6 format.
Multiline String	Holds a string of text. When this type of string is displayed in a CentraSite user interface, the string is displayed in a multi-line text box and lines of text are wrapped to fit the width of the box. (Compare this with the String data type described in this table.)

Data Type	Description
	The **Internationalized** option allows you to store the text in internationalized string format.
Number	Holds a numeric value. When you define an attribute of this type, you can specify the number of decimal positions that are to be shown when the attribute is displayed in a user interface. If you do not want to restrict the number of decimal positions that the user interface displays, choose the **Maximum Precision** option to display all positions. You can optionally assign a label such as `Seconds`, `tps`, `KB`, `EUR` or `$` to attributes of this type, and specify whether this label is to appear as a prefix or a suffix when the attribute's value is displayed in a user interface. **Note:** The underlying data type for this kind of attribute is a Java double.
String	Holds a string of text. When this type of string is displayed in a CentraSite user interface, it is displayed in a single-line text box. If a value exceeds the width of the box, the excess characters are simply not displayed. The **Internationalized** option allows you to create a String attribute that holds different values for different locales. In CentraSite Control, for example, if a user logs on to CentraSite in an English locale and he or she assigns a value to an Internationalized String attribute, that value will be visible to other users with English locales. If a user in a German locale were to view the attribute, the attribute would appear empty because it has no value for the German locale. If the German-locale user were to subsequently assign a value to the attribute, the attribute would then have two String values: one in English and one in German. When CentraSite Control displays an Internationalized String, it displays the value associated with the user's current locale. In the example described above, it would show the English value to users with English locales and the German value to users in German locales. Users in other locales would see an empty attribute until a value for their locale had been assigned to the attribute. The **Enumeration** option allows you to specify a list of allowed values for the attribute.

Data Type	Description
URL/URI	Holds a URL/URI. This type of attribute only accepts values in the form: *protocol* ://*host* / *path* Where: ■ *protocol* is any protocol that java.net.URL supports ■ *host* is the name or IP address of a host machine ■ *path* (optional) is the path to the requested resource on the specified host
Relationship	Holds references to other registry objects. You use this type of attribute to express a relationship between an asset and another object in the registry.

Attribute Names

An attribute that is one of the following types has two names associated with it: a *display name* and a *schema name*.

Boolean

Date

Duration

Email

Multiline String

IP Address

Number

String

URL

The display name for these types of attributes is the name that is displayed by the CentraSite Control and CentraSite plug-in for Eclipse user interfaces. An attribute's display name can consist of any combination of characters, including spaces.

The following are all valid display names:

```
Business Owner
Amount (in $)
Numéro de téléphone
Avg. Invocations/Minute
1099 Code
```

You can change an attribute's display name at any time.

The attribute's schema name is the name that CentraSite actually gives to the underlying JAXR-based slot that represents the attribute in the registry. This name must be NCName-conformant, meaning that:

- The name must begin with a letter or the underscore character (_).
- The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals).
- The name cannot contain any spaces.

If you do not specify a schema name for an attribute, CentraSite automatically generates a default schema name based on the attribute's display name. It does this by taking the attribute's display name and replacing any spaces in the name with underscore characters (_) and/or by removing any invalid character in the name.

If you explicitly specify a schema name that is not NCName-conformant, CentraSite will request that you change it to an NCName-conformant name.

The following table describes the default schema names that CentraSite would generate for the display names shown above.

For this Display Name...	CentraSite would generate this schema name...	The resulting schema name is...
Business Owner	Business_Owner	*Valid.* You would not need to change the schema name.
Amount (in $)	Amount_in_	*Valid.* You would not need to change the schema name.
Numéro de téléphone	Numéro_de_téléphone	*Valid.* You would not need to change the schema name.
Avg. Invocations Minute	Avg._Invocations_Minute	*Valid.* You would not need to change the schema name.
1099 Code	_099_Code	*Valid.* You would not need to change the schema name.

Note also that an attribute's schema name must be unique within the type (i.e., two attributes in a type cannot have the same schema name).

After the attribute is created, you can no longer change its schema name.

Names for the Other Attribute Types

The concept of display names and schema names does not apply to the following types of attributes:

Classification

File

Relationship

These types are not represented using JAXR-based slots and, therefore, do not require underlying schema names. These attributes have one name, which can consist of any combination of letters, numbers, or special characters (including spaces).

Computed Attributes

CentraSite offers you the possibility to add computed attributes into asset type definitions and profiles; this allows you to define attributes which require complex computation in Java and then implement them as a Java plug-in, thus overcoming the limitations of predefined attribute types. You could, for example, make attribute values localizable by using computed attributes.

A computed attribute must describe its scale for the rendering within the profile of the asset type.

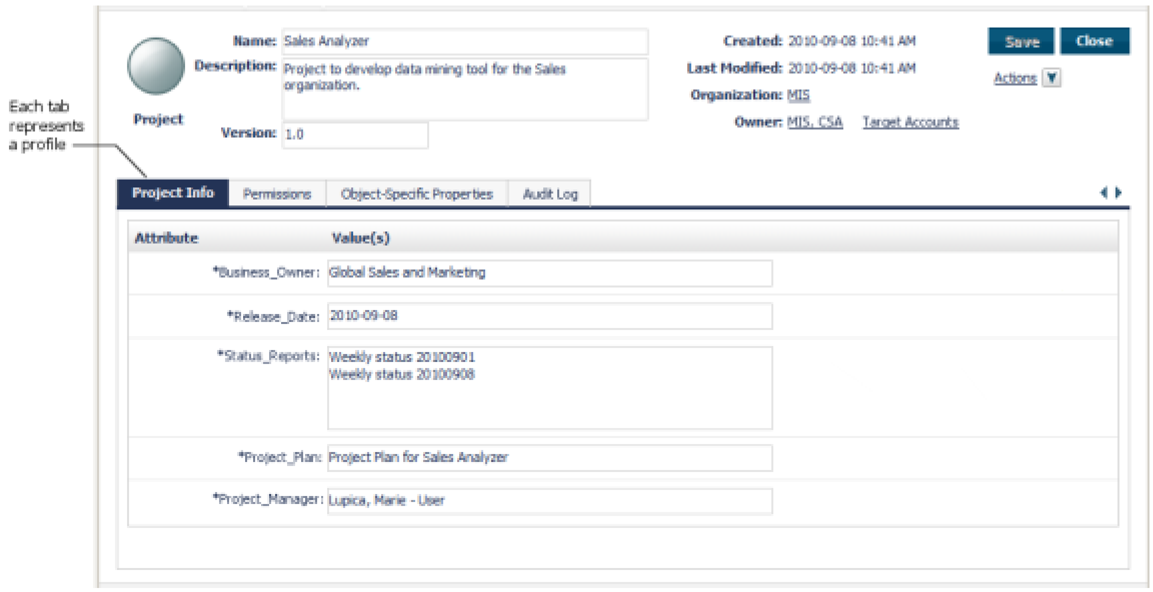
For a Java-based plug-in for a computed attribute, you create a jar file that contains the plug-in definition, and you load the jar file via CentraSite Control into the repository.

After you have added a computed attribute into a profile definition, you can perform administration tasks on the computed attribute in the same way as for normal attributes. For example, you can define the ordering of the attributes in a profile, regardless of whether they are standard attributes or computed attributes.

Profiles

When a type defines an asset (that is, if it is an asset type), the attributes that make up the type are assigned to profiles. Profiles determine how the type's attributes are grouped and presented when an instance of that type is displayed in the CentraSite graphical user interface. In CentraSite Control, for example, the attributes associated with a particular profile are grouped together on a tab.

In CentraSite Control, profiles appear as tabs



Name: Sales Analyzer
Description: Project to develop data mining tool for the Sales organization.
Version: 1.0
Created: 2010-09-08 10:41 AM
Last Modified: 2010-09-08 10:41 AM
Organization: MIS
Owner: MIS, CSA Target Accounts

Attribute	Value(s)
*Business_Owner:	Global Sales and Marketing
*Release_Date:	2010-09-08
*Status_Reports:	Weekly status 20100901 Weekly status 20100908
*Project_Plan:	Project Plan for Sales Analyzer
*Project_Manager:	Lupica, Marie - User

When you define an asset type, you specify the profiles on which its attributes are to be displayed. CentraSite does not require an attribute to be assigned to a profile. If you do not assign an attribute to a profile, the attribute will not be visible in the user interface. However, the attribute will reside in the type definition. You can assign an attribute to multiple profiles if you want it to appear on multiple profiles (tabs) in the user interface.

You can define any number of profiles for an asset type. You can specify the order in which you want the profiles to appear when an instance of the type is displayed. You can also specify the order in which attributes are to be displayed within each profile.

Generic Profiles

In addition to the profiles that you define, CentraSite provides several predefined profiles, called *generic profiles*, which you can optionally include in an asset type.

The generic profiles contain system-defined attributes and controls. The information on the generic profiles is generated by CentraSite. You cannot customize the content of the generic profiles or add attributes to them. You can, however, select which of these profiles you want CentraSite to include with an asset type.

Generic Profile	Description
Summary	*Available only for Service, XML Schema, Application Server, BPEL Process, Interface and Operation asset types.* Provides basic information about the asset. For a service asset, this profile includes a list of the operations and bindings that the service provides. For a BPEL Process, Interface, or Operation asset, the

Generic Profile	Description
	profile shows basic information such as the asset owner, but includes no type-specific attributes.
Consumers	Displays the list of users and applications that are registered to consume the asset. This profile also contains controls for registering an application, a user, or a group as a consumer of the asset.
Subscriptions	Displays the list of users who are registered to receive notifications when changes are made to the asset.
Classification	Displays the list of categories by which an asset is classified. This profile also contains controls for adding `ad hoc` classifiers to an asset.
Associations	Displays the list of objects to which the asset is related. This profile also contains controls for establishing "ad hoc" relationships between an asset and other objects in the registry.
Permissions	Displays the asset's instance-level permissions. This profile also contains controls for modifying the asset's instance-level permissions. To view all of the instance-level permissions for an asset, a user must have Modify or Full permissions on the asset. To edit the instance-level permissions for an asset, a user must have Full permissions on the asset. If a user has only View permission on an asset, the Permissions profile shows only that particular user's permissions for the asset.
Versions	Displays the versioning history for an asset and provides links to earlier versions and revisions of the asset. This profile also contains the controls for generating a new version of the asset, purging older versions of the asset and reverting to a previous version of an asset.
External Links	Displays the list of links to external documents and files that are attached to the asset. This profile also contains controls for attaching documents and files to an asset.
Object-Specific Properties	Displays the list of object-specific properties that have been assigned to an asset. An object-specific property consists of a key, which identifies the name of the property, and an

Generic Profile	Description
	optional String value, which contains the data associated with the property. (A property's value can be null.) Object-specific properties are used to hold information about an instance of an asset when there is no predefined attribute to hold that data. Typically, they are used in one-off situations to attach ad-hoc data to an instance of an asset. For example, if you were managing a certification effort, you might use an object-specific property to identify the set of assets that required certification.
Audit Log	Displays the update activity associated with an asset. This profile lists each change that has been made to an asset (including changes in an asset's lifecycle state) and identifies the user that made the change.
Members	*Available for only Package and IS Package types.* Displays the list of objects to which the asset is related. This profile also contains controls for establishing "ad hoc" relationships between an asset and other objects (for example, assets, users or organizations) in the registry.
Performance	Displays the run-time performance metrics associated with an asset. If you are using webMethods Mediator, webMethods Insight, or another run-time monitoring component to log performance metrics for an asset, CentraSite displays those metrics on this profile. **Note:** The **Performance** profile is displayed for virtual services regardless of whether you enable or disable it for the Service asset type. That is, when you disable the **Performance** profile for the Service asset type, CentraSite removes the profile from the service assets, but continues to display it for virtual services.
Events	Displays the run-time events associated with an asset. If you are using webMethods Mediator, webMethods Insight, or another run-time monitoring component to log run-time events for an asset, CentraSite displays those events on this profile. **Note:** The **Events** profile is displayed for virtual services regardless of whether you enable or disable it for the Service asset type. That is, when you disable the **Events** profile for the Service asset type, CentraSite removes the profile from service assets, but continues to display it for virtual services.

Generic Profile	Description
Policies	Displays the list of design/change-time policies and run-time policies that are applicable to an asset (that is, it displays all the policies whose scope encompasses the displayed asset). **Note:** The **Policies** profile is displayed for virtual services regardless of whether you enable or disable it for the Service asset type. That is, when you disable the **Policies** profile for the Service asset type, CentraSite removes the profile from service assets, but continues to display it for virtual services.
Processing Steps	*Available for Virtual Service, Virtual XML Service, and Virtual REST Service types.* For a Virtual Service, displays the protocol (HTTP, HTTPS or JMS) and SOAP format (1.1 or 1.2) of the requests that the service will accept. For a Virtual REST Service or Virtual XML Service, displays the protocol (HTTP or HTTPS) of the requests that the service will accept. Also, you specify the HTTP methods (GET, POST, PUT, DELETE or Use Context Variable) that are supported by the native service. This profile also contains the request routing methods and protocol for authenticating the requests.
Deployment	*Available for Virtual Service, Virtual XML Service and Virtual REST Service types.* Displays the list of virtualized services that are ready for deploying in the webMethods Mediator target.

Computed Profiles

CentraSite offers you the possibility to add computed profiles into asset type definitions; this gives you the option to define your own profile, which means that you can implement your own algorithms for calculating the values you wish to represent. You could for example aggregate or compute attribute information from embedded or linked objects.

You can combine the attribute specific profile and the generic profiles layout concept in a single computed profile.

Computed profiles let you create your own layout by using a UI Rendering Concept. You can also specify your own rendering logic to display the computed values. You could, for example, create a custom display of performance metrics as a graphic or an animation.

A computed profile can be implemented as a Java plug-in. For a Java-based plug-in for a computed profile, you create an archive file that contains the plug-in definition, and you load the archive file via CentraSite Control into the repository.

After you have added a computed profile into the asset type definition, you can perform administration tasks on the computed profile in the same way as for normal profiles. For example, you can define profile-based permissions, and you can define the order of the computed profile relative to the other profiles in the asset detail display.

Assigning Permissions on Profiles

You can restrict access to individual profiles by setting profile permissions on an instance of an asset. Doing this enables you to control who can view and/or edit the attribute values on a particular instance of a profile. For more information about controlling access to individual profiles within an asset, see the *CentraSite User's Guide*.

Important: Profile permissions restrict access at the UI level but not the API level. At the API level, profile permissions are irrelevant. If a user has view permission on an asset, he or she can access all of the asset's metadata through the API, regardless of whether profile permissions exist for the asset.

Who Can Create and Manage Types?

To create custom types, you must belong to a role that has the Manage Asset Types permission. Besides allowing you to create custom types, this permission allows you to edit and delete any user-defined asset type. Additionally, it allows you to edit certain predefined types that are installed with CentraSite. By default, users in the CentraSiteAdministrator or Asset Type Administrator role have this permission, although an administrator can grant this permission to other roles.

For more information about roles and permissions, see "About Roles and Permissions" on page 115.

Creating a New Type

CentraSite provides a wizard for defining new types.

Before You Begin

Before you begin to create a new type, keep in mind the following points:

- A type has two names: a *display name* and a *schema name*.
  - The display name is the name that CentraSite uses when it refers to the type in the user interface. (This is the name that appears in the catalog browser, for example.) The display name can contain any character, including spaces.

  - The schema name is the name that is given to the underlying schema that contains the type definition. The schema name must conform to the naming requirements for an NCName data type, which does not permit names with spaces or most special characters.

  By default, CentraSite derives the schema name from the display name that you specify for the type. If your display name includes special characters, then you must explicitly specify a schema name that is NCName conformant in the type's **Advanced Settings** dialog box.

- If you intend to include a Classification attribute in your custom type, make sure that the corresponding taxonomy exists before you begin creating the new type. To add a Classification attribute to a type, you must specify the taxonomy with which the attribute is associated. You cannot do this unless the taxonomy already exists in CentraSite.
- If you intend to include a Relationship attribute in your custom type, make sure that the corresponding association type exists before you begin creating the new type. To add a Relationship attribute to a type, you must specify the type of association that the attribute represents. You cannot do this unless the association type is already defined CentraSite. For information about defining association types, see "Working with Association Types" on page 302.

Creating a New Type

You use the **Add Asset Type** wizard to specify the properties and options for a new asset type.

To create a new type

1. In CentraSite Control, go to **Administration > Types**.
2. On the **Types** tab, click **Add Asset Type** to open the **Add Asset Type** wizard.
3. In panel 1, complete the following fields:

In this field...	Do the following...
Name	Enter a display name for the type. Choose a name that your users will recognize and understand. For example, use `BPEL Process Document`, not `bpdoc`. The name you assign to the asset type can contain any character, including spaces. However, if you specify a name that does not conform to the NCName type, you *must* click **Advanced Settings** and specify a name that is NCName conformant in the **Schema Name** field. **Note:** If the name that you assign to the asset type is NCName conformant, except that is includes spaces, it is

In this field...	Do the following...
	not necessary to explicitly specify the type's schema name. CentraSite automatically replaces space characters with the _ character when it generates the schema name for an asset type.
Description	*Optional*. Enter a brief description of the type.

4. Click **Advanced Settings** and complete the following steps as necessary.

   a. Verify that the schema name and the namespace name that were generated by CentraSite are valid.

In this field...	Do the following...
Schema Name	Verify that the schema name that CentraSite generated for this asset type is NCName conformant. CentraSite automatically populates this field with a name that is derived from the asset type name that you specified in the previous step. For example, if your asset type name is `My Asset Name`, CentraSite automatically populates this field with `My_Asset_Name`. If the schema name generated by CentraSite does not meet the following criteria, you must specify a name that does. ■ The name must begin with a letter or the underscore character (_). ■ The remainder of the name can contain any combination of letters, digits, or the following characters: . - _ (i.e., period, dash, or underscore). It can also contain combining characters and extender characters (e.g., diacriticals). ■ The name cannot contain any spaces. Additionally, the type's fully qualified schema name must be unique among all types in the registry. **Note:** You cannot change the schema name after the type is created.
Namespace	Modify the namespace that CentraSite has proposed for the type if necessary. By default, CentraSite generates the namespace in the following form: `http://namespaces.`*OrganizationName*`.com/Schema`

In this field...	Do the following...
	Where, *OrganizationName* is the name of your organization. The **Namespace** value is used to qualify the name specified in **Schema Name**. Together, the **Schema Name** and **Namespace** values produce the type's fully qualified name. This name must be unique within the registry. You generally do not need to modify the namespace that CentraSite proposes for a type. In most cases, the proposed namespace will be adequate. However, you might modify the namespace if you want it to include the name of a different organization or if you need to resolve a naming conflict between this type and an existing type. **Note:** You cannot change the namespace value after the type is created.

b. If you want to use a custom icon to represent this type in the user interface, upload large and small versions of the icon as described below.

Note: If you do not specify a custom icon, CentraSite assigns the default icon to the type.

In this field...	Do the following...
Large Icon	*Optional*. Specify the large icon that is to be used to represent this type. CentraSite Control uses this icon when it displays the details for an instance of the type. If you want to use a custom icon, click the **Browse** button and upload the file containing the large version of the icon to CentraSite. The icon must be in GIF format. To ensure proper alignment when it is displayed in the user interface, the icon must be 64 x 64 pixels in size.
Small Icon	*Optional*. Specify the small icon that is to be used to represent this type. CentraSite Control displays this icon when instances of the type appear in lists or summary tables. If you want to use a custom icon, click the **Browse** button and upload the file containing the small version of the icon to CentraSite. The icon must be in GIF format. To

In this field...	Do the following...
	ensure proper alignment when it is displayed in the user interface, the icon must be 16 x 16 pixels in size.

c. Specify the type's advanced options as follows:

Enable this option...	To...
Visible in Asset Browse	Allow instances of this type to be displayed in the catalog browser. When you enable this option, CentraSite Control includes the type in the **Asset Types** pane on the **Asset Catalog > Browse** page. When you disable this option, CentraSite Control omits the type from the **Asset Types** pane so users cannot browse for instances of the type.
Enable reports	Allow reports to be generated against instances of this type.
Policies can be applied	Allow user-defined design/change-time policies to be created and enforced for instances of this type. **Note:** If you disable this option, CentraSite will not apply any user-defined design/change-time polices to instances of the type, even in cases where the policy is designed to execute against *all* asset types. **Note:** This option does not apply to system policies. CentraSite will apply system policies to instances of the type whether this option is enabled or not.
Require Consumer Registration	Require users to register an application when they submit consumer registration requests for assets of this type.
Enable versioning	Allow users to generate versions of instances of this type. When you disable this option, CentraSite disables the **Add New Versions** command and omits the **Versions** profile from instances of this type.
Top Level type	Allow users to create instances of this asset type "from scratch." When you enable this option, users are allowed to create instances of the type using the **Add Asset** button in CentraSite Control.

Enable this option...	To...
	Generally you disable this option for types that are constituents of other assets, or for types that are only meant to be added to the registry by an importer. For example, the Operation type is used to represent an operation that belongs to a Web service. Operations are derived automatically from the service WSDL. They are not intended to be manually defined by users. Therefore, the Operation type is not designated as a "top level type".
Enable Lifecycle Management	Allow a lifecycle model to be applied to assets of this type.
Visible in Search	Allow users to define a search for this particular type. When you enable this option, CentraSite Control includes the type in the **Types** list on the Advanced Search page. Including a type in this list enables users to define queries that select on that specific type. **Note:** If you change the state of this option and then do an advanced search, you might need to refresh the Advanced Search page to see the change reflected in the **Types** list.

d. Click **OK**.

5. In panel 2, specify the attributes that make up the type. For detailed instructions. If you need procedures for this step, see "Define Attributes for a Type" on page 241.
6. In panel 3, define profiles for the type and assign attributes to them. If you need procedures for this step, see "Define Profiles for an Asset Type" on page 250.
7. In panel 4, specify which of the generic profiles you want to include with the type. If you need procedures for this step, see "Generic Profiles" on page 231.
8. In panel 5, specify the order in which you want the profiles to appear when they are displayed in the user interface.
9. Click **Finish** to save the new type.

Define Attributes for a Type

General Procedure

Use the following steps to complete panel 2 in the **Add Asset Type** or **Edit Asset Type** wizard. You use this panel to define the set of attributes that make up the asset type.

Using the Dialog

To define attributes for an asset type

1. Go to panel 2 in the **Add Asset Type** wizard (if you are creating a new type) or the **Edit Asset Type** wizard (if you are modifying an existing type). If you need procedures for this step, see "Creating a New Type" on page 236 or "Viewing or Editing a Type" on page 256.
2. To add a new attribute to the type, click **Add Attribute**.

   Complete the following fields in the **Add Attribute** dialog box:

In this field...	Specify...
Name	The display name for the attribute. This is the attribute name that users will see when they view instances of this type in the user interface, therefore, the name should be meaningful. The display name can contain any combination of characters, including spaces. You can change an attribute's display name at any time. **Note:** If you are defining a Relationship attribute, by default the attribute's name will be derived from the name of the association type that you assign to the attribute. You can, however, assign a custom name to the Relationship attribute by specifying the **Name** attribute.
Schema Name	The schema name for the attribute. The schema name is the internal name that CentraSite assigns to the attribute. This name can contain only letters, numbers and the underscore character (_). If you do not enter a schema name, CentraSite automatically generates a schema name for the attribute based on the name you enter in the **Name** field. However, if the name that CentraSite generates includes invalid characters, you will be prompted to provide a valid schema name when you save the attribute. For information about valid schema names, see "Attribute Names" on page 228.

In this field...	Specify...
	After you create and save the attribute, you can no longer change its schema name. **Note:** Relationship, File, and Classification attributes do not require schema names.
Description	*Optional.* A brief description for the attribute.
Required	Whether the attribute is required or optional. When you enable this option, CentraSite prevents users from saving an asset of this type without first assigning a value to this attribute. In the user interface, a required attribute is displayed with an asterisk (*) next to its name. **Note:** An attribute can be a required attribute and have a default value. If you do not supply a value for an attribute that is required and has a default value, the default value is automatically assigned to this attribute. When editing an existing type: ■ If there are no instances of the type in the registry, you can add a required attribute to an existing type. ■ If there are instances of the type in the registry, you can add a required attribute of type `slot` or `classification`, but not of type `relationship` or `file`. ■ If you add a new required attribute, no automatic update of existing instances takes place. This prevents the potential degradation of performance that could arise from the automatic update of a large number of instances. ■ You can enable the **Required** option for an existing attribute even if there are empty instances of the attribute. But in this case a default value must be provided for the attribute. ■ You can disable the **Required** option for an attribute at any time (even if assigned instances of the attribute already exist in the registry).
Read-only	Whether users can modify the value of the attribute. The **Read-only** option prevents the attribute's value from being changed after an asset is created. When you enable this option, users can assign a value to the attribute when an asset

In this field...	Specify...
	of this type is initially created. However, they cannot modify the attribute after the asset has been saved. You can enable or disable the **Read-only** option for an attribute at any time (even if instances of the type already exist in the registry).
Multiplicity	Whether the attribute can hold a just a single value or multiple values (i.e., an array of values). Enabling the **Multi Value** option allows users to assign more than one value to the attribute. **Note:** The **Multiplicity** option is not available for the Boolean attribute type. When editing an existing asset type: ■ You can switch an attribute's **Multiplicity** option from **Single Value** to **Multi Value** at any time (even if instances of the type already exist in the registry). ■ You can switch an attribute's **Multiplicity** option from **Multi Value** to **Single Value** only if: ■ No instances of the type exist in the registry. —OR— ■ Instances of the type exist in the registry, but each instance has at most one value assigned to this attribute. (i.e., no instances exist wherein this attribute has multiple values).
Data Type	The attribute's type. Be aware that after you create an attribute, you cannot change its type. Depending on the data type you select, additional type-specific fields or check boxes now appear in the dialog. Provide the appropriate information for these additional fields or check boxes. For details, see "Defining Data Types" on page 245.
Default Value	*Optional*. The value that is to be assigned to the attribute by default. **Note:** The **Default Value** option is not available for all attribute types. When editing an existing asset type:

In this field...	Specify...
	■ You can change a default value, assign a default value, or remove a default value from an attribute at any time. Changing the attribute's default value *does not* immediately affect any existing instances of the attribute. ■ If you add a default value to an attribute that did not previously have one, and the registry contains empty instances of that attribute, the default value will be assigned to those assets the next time that they are saved to the registry. ■ If you add a new attribute to an existing type and you assign a default value to that attribute, the default value will be assigned to the existing instances of that type the next time those instances are saved to the registry. If the attribute has the **Required** option set, then the following conditions apply: ■ When you create a new asset type with one or more required attributes, you do not need to provide a default value for these attributes during the creation process. ■ If instances of an asset type exist, and you update the type definition in any manner, regardless of whether you edit the required attributes or not (for example, adding a new optional attribute, adding a new required attribute, or even editing the type name), you must provide a default value for each required attribute of that type. Required attributes that have no value will be set to the default value the next time the instance is saved to the registry. ■ If instances with missing required attributes are viewed in CentraSite Control, these attributes are simulated and displayed with the default value. But the default value will not be added to the instance until the next save of the instance.

For the **Data Type** field, complete the type-specific options as described in "Defining Data Types" on page 245.

Click **OK** to save the attribute.

3. Repeat step 2 for each attribute that you want to add to the asset type.
4. If you need to edit an attribute that you have added to the asset type, click the name of the attribute in panel 2 and make your changes in the **Edit Attribute** dialog box. Be aware that certain options for an attribute cannot be changed after the attribute is created.
5. If you need to delete an attribute, select the attribute in panel 2 and click **Delete**.

> **Important:** CentraSite will not allow you to delete an attribute if an assigned instance of the attribute is present in the registry.

Defining Data Types

For the **Data Type** field, complete the type-specific options:

If you are adding this type of data type...	Do the following...
Classification	In the **Taxonomy** field, specify the taxonomy by which users will classify the asset. **Note:** After you create a classification attribute, neither its **Taxonomy** option or its **Default** option can be changed.
Computed Attribute	When you choose this data type, you are indicating that the attribute value, and the way it is rendered in a profile, will be generated using a Java plug-in. Choose **Java** as the implementation type, to indicate that you are using a Java plug-in. For more information, see "Defining a Computed Attribute" on page 247. In the field **File**, specify the name of the archive file that contains the Java plug-in. **Note:** When you define a computed attribute, the dialog fields **Required**, **Read-Only** and **Multiplicity** are not relevant, so they are automatically removed from the dialog.
Date/Time	In the **Display As** field, specify whether you want the user to specify: ■ The date and time ■ The date only ■ The time only
Relationship	Using this data type, you specify that an asset of the type you are currently defining can be related to another asset or object. You define the nature of the relationship in the **Relationship Type** field, using one of the following options: ■ Association

If you are adding this type of data type...	Do the following...
	■ Aggregation using Source ■ Aggregation using Target ■ Composition using Source ■ Composition using Target These options allow you to define a loose or tight coupling between the related objects. Depending on the type of relationship, operations on an asset (such as delete, export, move to another organization, set instance-level permissions, create a new version) can be applied automatically also to the related asset or object. For information about how you can use these relationship types, see "Working with Composite Types" on page 269. In the **Association Type** field, specify the type of association that this attribute represents. This is a label that you assign as a meaningful name for the relationship. Examples of labels are `hasChild` and `hasParent`, indicating a hierarchical relationship. In the **Relates To** field, specify the type(s) of object that can be the target of the relationship. You can specify more than one type of object by using the **+** icon. If you specify more than one type of object in this field, this means that an object of the current object type can have a relationship to an object that belongs to any of the given object types. **Note:** The **Relates To** option is enforced in the user interface and at the API level. Within the user interface, this option determines which types of object are shown to users when they set the attribute. At the API level, this option causes CentraSite to reject any asset whose attribute specifies an object that is not of an allowed type. After you create a Relationship attribute, you can modify the attribute's **Association Type***only* if there are no assigned instances of the attribute in the registry. Similarly, you can change or delete categories in the **Relates To** option *only* if the registry contains no assigned instances of the attribute. You can, however, assign additional categories to the **Relates To** option at any time (even if assigned instances of the attribute exist in the registry).

If you are adding this type of data type...	Do the following...
	For more information about association types, see "Working with Association Types" on page 302.
String or Multiline String	Enable the **Internationalized** option if you want to store the string values in internationalized string form. For more information about the Internationalized option, see the description of the String attribute in "Attribute Data Types" on page 225. After you create a String or Multiline String attribute, you cannot change the attribute's **Internationalized** option.
String	Enable the **Enumeration** option if you want to restrict the attribute to a predefined set of values. When users edit this attribute in CentraSite Control, the enumerated values are presented to them in a drop-down list. CentraSite also enforces the **Enumeration** option at the API level, meaning that it will reject any asset whose attribute does not contain one of the enumerated values. If you enable the Enumeration option for a String attribute, you can add new items to the enumerated list at any time (even if instances of the attribute exist in the registry). However, you can remove an item from the list only if that item is not currently assigned to an instance of this type in the registry. After you create a String attribute, you cannot switch it from an enumerated String to a non-enumerated String (or vice-versa).

Defining a Computed Attribute

The computed attribute is defined as an implementation of the `ComputedAttribute` interface. This section describes how to define a computed attribute.

Implementation Guidelines for Java-Based Computed Attributes

This section describes the Java interfaces and methods that you need to implement for a computed attribute.

Interfaces	Description
ComputedAttribute	This interface declares basic rendering methods for the user interface. It extends the ProfileAttribute interface. boolean: isUsed(): returns true if this attribute is used by at least one instance of the corresponding asset type. Collection: setValue(Collection): sets the value of this attribute. AttributeDescriptor: getAttributeDescriptor(): returns the definition of this attribute. String: getName(): returns the JAXR-based name of this attribute. Collection: getValue(): returns the value of this attribute. init(): adds the required parameters to the objects argument for attribute initialization. Collection<Concept>: getTargetObjectTypes(): returns the appropriate target registry object of this attribute. This method will be used when the attribute is of the relationship data type. Concept: getAssociationType(): returns the appropriate association type of this attribute. This method will be used when the attribute is of the relationship data type. ClassificationScheme: getTaxonomy(): returns the appropriate classification scheme of this attribute. This method will be used when the attribute is of the classification data type.

ProfileAttribute	void: init(Collection<CentraSiteRegistryObject>, Locale)
	
	Collection: getValue();
	
	String: getAttributeKey();
	
	AttributeDescriptor: getAttributeDescriptor();

AttributeDescriptor	This interface deals with the standard properties (isReadOnly(), is Required()...) for an attribute. String: getDataType(): returns the data type of this attribute. This can be one of the supported standard types (xs:...), or one of the special types: File, RichText, Image (allows the specification of the image type).

Interfaces	Description
	Object: getDefaultValue(): returns the default value of this attribute. String[]: getEnumValues(): int: getMaxLength(): returns the maximum length of this attribute. String: getMaxOccurs(): returns whether the attribute can have multiple occurrences. String: getMinOccurs(): returns whether the attribute is optional or required. Object: getNativeAtribute(): returns the native attribute instance if the data type is not primitive, otherwise returns null. int: getPrecision(): gets the precision for a `number` attribute. String: getUnitLabel(): returns the slot's unit label. boolean: hasDefaultValue(): returns whether this attribute has a default value. boolean: isPrefix(): Returns true if the slot's unit label is a prefix, and false if it is a suffix. boolean: isReadOnly(): returns whether this attribute is read-only. boolean: isRequired(): returns whether this attribute is required.

Structure of Archive File

The plug-in uses the policy engine infrastructure, so it uses the structure of a Java policy (zip file structure and classloader). The archive file must contain the following folders and files:

Zip folder	Description
META-INF	This folder contains the config.properties file, which is the build file for the plug-in. This properties file contains an entry of the following format: `com.softwareag.centrasite.computed.attr.impl.class= com.sample.StringAttrImpl`

Zip folder	Description
lib	This folder contains the archive file with the source code examples, the plug-in's executor class and the external libraries.

Sample Code

Sample Java code for implementing a computed attribute is supplied in the demos folder under CentraSite installation folder. The sample code is available in the files:

- ComputedAttribute.java
- ProfileAttribute.java
- AttributeDescriptor.java

Loading a Computed Attribute into an Asset Type Definition

After you have created an archive file that contains the attribute definition, you need to load the archive file into the asset type definition. You do this by starting the **Edit Asset Type** wizard for the appropriate existing asset type, or the **Add Asset Type** wizard for a new custom asset type, and specifying in the wizard that you are defining a new computed attribute.

For details on how to load an archive file of a computed attribute into an asset type definition, see "Define Attributes for a Type" on page 241.

When you have loaded the archive file, the new attribute is displayed in the list of attributes that can be assigned to an asset type profile.

Define Profiles for an Asset Type

General Procedure

Use the following procedure to complete panel 3 in the **Add Asset Type** or **Edit Asset Type** wizard. You use this panel to define profiles and their attributes.

You can add a profile in two ways:

- **Manually, using the asset type's available attributes.** To define a profile manually, you select attributes from the list of available attributes and assign them to the profile.
- **Using a Java plug-in that contains a computed profile.** A computed profile is a user-defined profile that is implemented as a Java plug-in. You add the computed profile to the asset type by importing the computed profile's definition from an archive file. The plug-in specifies the attributes that are contained in the profile. The plug-in also has the sole responsibility for rendering the layout representation within the profile. After a computed profile has been defined for an asset type, the computed profile is treated in the same way as any other profile; for example, permissions for computed profiles can be granted in the same way as for standard profiles, and the ordering

of profiles within a type definition is the same for computed profiles as for standard profiles.

To add a profile to the asset type

1. Go to panel 3 in the **Add Asset Type** wizard (if you are creating a new type) or the **Edit Asset Type** wizard (if you are modifying an existing type). If you need procedures for this step, see "Creating a New Type" on page 236 or "Viewing or Editing a Type" on page 256.
2. To add a new profile to the type, click **Add Profile** and perform the following steps.
   a. In the **Profile Name** field, enter a name for the profile. This is the name that users will see in the user interface when they view an asset of this type. Therefore, the name you assign should be meaningful to your users (for example, `Technical Notes`, not `tn`).

      Note: You cannot give a profile the same name as any of the generic profiles as described in "Generic Profiles" on page 231.

   b. If the profile you wish to add is a computed profile, mark the checkbox **Computed Profile** and select the implementation language from the drop-down list. If, for example, you have implemented the computed profile using Java, then select **Java**.

      When you mark the checkbox, the dialog's list of available attributes is replaced by the input field **Profile Implementation Archive**. In this field, specify the name of the archive file that contains the definition of the computed profile, then click **OK**.

      For more information about how to implement a computed profile, see "Defining a Computed Profile" on page 252.

      Note: If you define a profile as a computed profile, you cannot change the profile back to being a non-computed profile at a later stage. You can, however, change the archive file that contains the definition of the computed profile.

   c. If your new profile is not a computed profile, continue with this step.

      Use the arrow buttons to specify the attributes for the new profile and the order in which the attributes will be displayed.

      - Add an attribute to the profile by marking the attribute's checkbox in the list of available attributes and clicking the right-pointing arrow.
      - Add all of the available attributes to the profile by clicking the right-pointing double arrow.
      - Remove an attribute from the profile by marking the attribute's checkbox in the list of assigned attributes and clicking the left-pointing arrow.
      - Remove all attributes from the profile by clicking the left-pointing double arrow.

- Change the position of an assigned attribute relative to the other assigned attributes within the profile by marking the checkbox of the attribute and clicking the up or down arrow as required.

d. Click **OK**.

3. If you need to edit a profile that already exists within the type, click the name of the profile in panel 3 and make your changes in the **Edit Profile** dialog box.

   Use the arrow buttons, as described in the previous step for **Add Profile**, to specify the attributes for the profile, and the order in which the attributes will be displayed.

4. If you need to delete a profile, select the profile in panel 3 and click **Delete**.

   Note: Deleting a profile *does not* delete the attributes that were assigned to the profile. It simply removes the profile definition from the asset type. The attributes still exist within the type. However, they will not be visible in the user interface unless you assign them to another profile.

Defining a Computed Profile

The following topics describe how to define a computed profile.

Definition of Java-Based Computed Profile

A Java based computed profile has the following rendering mechanism:

- WithUiRendering: This dictates the user-defined rendering of the profile's attributes.
- WithoutUiRendering: This dictates the CentraSite's default rendering of the profile’s attributes. This default rendering is based on the attribute’s data type.

Implementation Guidelines for Java-Based Computed Profiles

Interfaces	Description
ComputedProfile	This interface declares basic rendering methods for the user interface. void: init(java.lang.Object, Locale): with `CentraSiteRegistryObject` as a parameter where the necessary implementation is done and updateAsset() which would return a collection of registry object serves as a save hook. boolean: canRenderUI(): This determines whether the rendering is based on the UI (true) or on the triples associated with the profile (false). Collection: getAttributes(): returns a collection of `ProfileAttribute` and would be called only when canRenderUI() returns true.

Interfaces	Description
	Collection: updateAsset(): returns a collection of `CentraSiteRegistryObject` and would be called only when canRenderUI() returns true.
WebUIProfile	This interface is specific to the rendering for CentraSite Control. java.lang.Object: createProfileContent(): create the profile contents with XML content (in compliance with Application Designer).
EclipseProfile	This interface is specific to the rendering for Eclipse Designer. createFormContent(): create the profile contents on the supplied IManagedForm (in compliance with Eclipse Designer SWT/JFace library).
BUIProfile	This interface is specific to the rendering for CentraSite Business UI. getEditPageURL(): returns URL of the edit page of computed profile and would be called only when canRenderUI() returns true. getViewPageURL(): returns URL of the view page of computed profile and would be called only when canRenderUI() returns true. getProfileDataAsJson(): returns a collection of profile data as JSON-formatted string. Collection computeProfileData (String userInputsAsJSON): sets a collection of profile data as JSON-formatted string.
ProfileAttribute	This interface deals with the standard UI rendering of the profile's attributes and allows defining the attributes as key value pairs. The rendering of each attribute will be the standard rendering for the corresponding datatype of the attribute. AttributeDescriptor: getAttributeDescriptor(): returns the descriptor of the attribute. String: getAttributeKey(): returns the key of the attribute. String: getName(): returns the name of the attribute.

Interfaces	Description
	Collection: getValue(): returns the value of the attribute.
ComputedAttributeLine	This interface deals with the user-defined UI rendering of the profile's attributes. The UI rendering of the attributes will be determined by the coding of this interface. void: buildUI(StringBuilder layout): This method is responsible for rendering the layout definition. void: passivate(): This method is responsible for storing the values back to the object. void: revert(): This method is to revert the changes.
AttributeDescriptor	This interface deals with the standard properties (isReadOnly(), is Required()...) for an attribute. String: getMinOccurs(): returns the minimum allowed occurrences of this attribute. String: getMaxOccurs(): returns the maximum allowed occurrences of this attribute. boolean: isRequired(): returns whether this attribute is required. boolean: isReadOnly(): returns whether this attribute is read-only. boolean: hasDefaultValue(): returns whether this attribute has a default value. Object: getDefaultValue(): returns the default value of this attribute. int: getMaxLength(): returns the maximum length of this attribute. String: getDataType(): returns the data type of this attribute. Object: getNativeAtribute(): returns the native attribute instance if the data type is not primitive, otherwise returns null. String[]: getEnumValues(): boolean: isPrefix(): returns whether this slot's unit label is a prefix or suffix value. Return `true` if the unit label is a prefix, `false` for suffix.

Interfaces	Description
	int getPrecision(): returns the precision for a `number` attribute. String: getUnitLabel(): returns this slot's unit label. The label may be null.

Structure of Archive File

The archive file must contain the following folders and files:

Zip folder	Description
META-INF	This folder contains the config.properties file, which is the build file for the plug-in. This properties file contains an entry of the following format: `com.softwareag.centrasite.computed.profile.webui.impl.class= com.sample.MyProfileImpl`
lib	This folder contains the archive file with the source code examples, the plug-in's executor class and the external libraries.

Sample Computed Profiles

Your CentraSite installation contains 5 sample computed profiles. Two samples are for the CentraSite Control, one for the CentraSite Business UI, and the other two are for the Eclipse Designer.

Sample Computed Profiles for CentraSite Control

Your CentraSite installation contains two sample computed profiles (which is contained in demos folder) that you can use to create an archive file for the computed profile specific to CentraSite Control.

- NonPrimitiveDataTypeSamples
- SampleComputedProfile

Sample Computed Profiles for Business UI

Your CentraSite installation contains a sample computed profile (which is contained in demos folder) that you can use to create an archive file for the computed profile specific to CentraSite Business UI.

- SampleProfile

Sample Computed Profiles for Eclipse Designer

Your CentraSite installation contains two sample computed profiles (which is contained in demos folder) that you can use to create an archive file for the computed profile specific to Eclipse Designer.

- DesignerProfileSample
- SampleComputedProfile

Load a Computed Profile into an Asset Type Definition

After you have created an archive file that contains the profile definition, you need to load the archive file into the asset type definition. You do this by starting the **Edit Asset Type** wizard for the appropriate asset type and specifying in the wizard that you are defining a new computed profile.

For details about how to load the archive file of a computed profile into an asset type definition, see "Define Profiles for an Asset Type" on page 250.

When you have loaded the archive file, the new profile is displayed in the detail page of all assets of the asset type.

Viewing or Editing a Type

You use the **Edit Asset Type** wizard to view or edit the attribute settings for an existing asset type.

Before You Begin

Before you view or edit a type, keep in mind the following points:

- You can modify the type's display name, description, icons and advanced options. You cannot modify the type's **Schema Name** or its **Namespace** property. These two properties are set when the type is created and cannot be changed thereafter.
- With respect to attributes:
  - You can add new optional attributes to the type at any time.
  - You can add new required attributes if there are no existing instances of the type in the registry. If there are existing instances of a type, you can add a new required attribute of type `slot` or `classification`, but not of type `relationship` or `file`.
  - If you add a new attribute, regardless of whether it is optional or required, no automatic update of existing instances takes place. This prevents the potential degradation of performance that could arise from the automatic update of a large number of instances.

  - You cannot modify the data type for an attribute, but you can modify many of an attribute's other options. Be aware that certain attributes are not permitted if assigned (that is, non-empty) instances of the attribute are present in the registry. For information about the kinds of changes you can make to an existing attribute, see the property and option descriptions in "Define Attributes for a Type" on page 241.
  - You cannot delete an attribute from a type if an instance of the type exists with a value assigned to the attribute. In such a situation, you must first remove the attribute's value from each such instance before you can delete the attribute.

    Note that the command line tool `CentraSiteCommand` provides support for removing an attribute, in cases where existing instances contain a value for the attribute.
  - A sub type inherits all of its attributes from its base type. To add new attributes to a sub type, you do one of the following:
    - Add the attributes to profiles of the base type and inherit the profiles in sub type, OR
    - Clone the profiles of the base type, and then add the attributes to the cloned profiles in sub type.

    You can selectively display these attributes on the profiles that you have defined in or cloned from the base type.
  - You cannot delete attributes or edit the properties of attributes within the inherited profiles in the sub type. However, you can delete and edit attributes within cloned profiles in the sub type.
- You can add, modify, delete, rename, and reorganize the profiles associated with a base type at any time.
- You can modify, delete, rename, and reorganize the base type profiles associated with a sub type at any time, if the profiles are cloned from the base type.
- You cannot modify, delete or rename the base type profiles associated with a sub type, if the profiles are inherited from the base type.
- Modifying the cloned profiles and attributes in a sub type will not affect the profiles and attributes in the base type.
- When recloning the base type profiles within a sub type, any new attribute that is added to the base type, and any attribute that is already deleted from the sub type but still available in the base type, will be cloned in the sub type.
- Recloning the base type profiles within a sub type does not affect the existing attributes in the sub type.

Important: If you are modifying one of the predefined asset types installed with CentraSite, review the information in "Predefined Asset Types in CentraSite" on page 263 before you begin. It explains the kinds of modifications that you can make to the predefined types.

Important: If you are using CentraSite in conjunction with other software products, for example, the products of the Software AGwebMethods Product Suite or a third-party product, those products can add their own asset types to CentraSite. Be aware that CentraSite treats these types as user-defined custom types, which can be modified by an administrator with the appropriate permissions (just like any other custom type). Modifying or deleting these types in CentraSite can lead to inconsistencies or errors in the product that uses the type. For example, if you modify or delete a type that is used by the webMethods Product Suite, components such as the webMethods Integration Server may no longer be able to publish assets to CentraSite. To prevent these types of errors, *do not modify or delete any asset type on which other Software AG components or third-party products depend*.

How to View or Edit an Asset Type

To view or edit a type, follow these steps:

To view or edit a type

1. In CentraSite Control, go to **Administration** > **Types**.

   By default, all of the available types are displayed in the **Types** tab.

2. If you want to filter the list to see subset of the available types, enter a partial string in the **Search** field. CentraSite applies the filter to the **Name** column. The **Search** field is a type-ahead field, so as soon as you enter any characters, the display will be updated to show only those types whose name contains the specified characters. The wildcard character `%` is supported.
3. In the **Types** tab, click the name of the asset type that you want to modify.
4. In the Asset Type Details page, click **Edit** to open the **Edit Asset Type** wizard.

   The **Edit Asset Type** wizard shows the current attribute settings for the selected asset type. You can edit the type's settings by typing or selecting the values you want to specify on each of the panels.

5. Click **Finish** to save the updated type.

How to View Multiple Asset Types

You can view multiple asset types as follows:

To view multiple asset types

1. In CentraSite Control, go to **Administration** > **Types**.
2. Ensure that the **Types** tab is selected.
3. Mark the checkboxes of the types whose details you want to view.
4. In the **Actions** menu, click **Details**.

The **Details** view of each of the selected types is now displayed.

Removing an Attribute Using the Command Line Tool

Note: When you remove an attribute from an asset type, keep the following in mind:

- If there are many existing instances, the remove operation can take some time to complete.
- You cannot remove a predefined attribute from a predefined asset type. You can, however, remove a custom (i.e. user-defined) attribute from a predefined asset type.

There is a command line tool `remove Attribute` that allows you to remove an attribute from an asset type, in cases where existing instances of the type contain a value for the attribute. Using the tool, the attribute's value is automatically removed from all existing instances, then the attribute is removed from the type.

You can use the command line tool `CentraSiteCommand.cmd` (on Windows) or `CentraSiteCommand.sh` (on UNIX). The command line tool is located in <*SuiteInstallDir* >/CentraSite/utilities.

If you start the command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

To remove an attribute from an asset type, use a command of the following format:

```
CentraSiteCommand remove Attribute [-url <CENTRASITE-URL>]
-user <USER-ID>-password <PASSWORD> -assetType <ASSET-TYPE>
[-attributeKind <ATTRIBUTE-KIND>] -attributeName <ATTRIBUTE-NAME>
```

Input Parameters

The following table describes the complete set of input parameters that you use with the `remove Attribute` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.

Parameter	Description
`-assetType`	The name of the asset type, in the format "{<namespace of the asset type>}SchemaName".
`-attributeKind`	A one-character code representing the type of the attribute you wish to remove. Allowable values are "C" for Classification, "R" for Relationship, "F" for File and "S" for all other attribute types. The use of the `attributeKind` parameter is optional. For more information about the various kinds of attributes, see "What Is a Type?" on page 220.
`-attributeName`	The name of the attribute schema for attributes whose *AttributeKind* is "S". For attributes with an *AttributeKind* other than "S" it is the name of the attribute itself.

Example

```
CentraSiteCommand remove Attribute -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -assetType
{http://namespaces.CentraSite.com/Schema}XMLSchema
-attributeKind S -attributeName test_String_Attribute
```

Configuring Inheritance for Base Type Profiles, LCMs, and Policies

The Inherit options for a sub type determines whether the profiles, lifecycle models, and policies that are associated with a base type also apply to its sub type. You can enable or disable this option for a sub type.

Inheriting Base Type Profiles to Sub Types

When an asset is an instance of a sub type, the set of profiles that CentraSite applies to the asset instance depends on the sub type's **Inherit base type profiles** setting.

When the **Inherit base type profiles** option is enabled for a sub type, CentraSite inherits the profiles of the base type in addition to the profiles of the sub type to the asset instance. For example, when you enable this option for the sub type REST Service, an asset instance of this type will include both the profiles that are defined for that sub type REST Service, and the profiles that are defined for its base type Service.

When this option is disabled for a sub type, CentraSite applies the profiles that are defined for that sub type; but it does not inherit the profiles of its base type. For example, when you disable this option for the sub type REST Service, asset instances of the REST

Service type will simply include profiles that are defined for that type. Asset instances will not inherit the profiles that are defined for its base type Service.

By default, the **Inherit base type profiles** option is disabled for a type.

The following table summarizes how the set of profiles that CentraSite applies for a type is affected by the state of the **Inherit base type profiles** option.

If the type's "Inherit Base Type Profiles" option is...	Instances of the type will have profiles of the...
ENABLED	Base Type —AND— Sub Type
DISABLED	Sub Type

To inherit the base type profiles in sub types

1. In CentraSite Control, go to **Administration** > **Types**.

   By default, all of the available types are displayed in the **Types** tab.
2. In the **Types** tab, click the name of the sub type whose base type profiles you want to inherit.
3. In the Asset Type Details page, click **Edit** to open the **Edit Asset Type** wizard.
4. On the **Edit Asset Type** wizard, click **Advanced Settings**.
5. Enable the **Inherit base type profiles** option.
6. Click **OK**, and then **Finish**.

   After you have inherited the base type profiles in the sub type, you can view the inherited profiles and its attributes. However, you cannot customize their contents.

Cloning Base Type Profiles in Sub Types

The Clone option for a sub type determines whether the profiles associated with the base type are also applied to the sub type. In addition, this option provides the ability to create, edit, and delete any number of attributes and profiles in a sub type, without affecting its base type.

If the **Inherit base type profiles** option is not already enabled in the **Advanced Settings** dialog of a sub type, CentraSite presents a **Clone Base Type Profiles** dialog that prompts for cloning the base type profiles. After confirming the cloning, the appropriate base type profiles will be cloned in the sub type.

This option lets you make a copy of the base type profiles in the sub type, and save it to the user defined profiles to enable certain permissions (view, create, edit and delete) for one or more users on an instance of that particular type.

Profile cloning enables you to customize the predefined profiles that are shipped with the base type.

To clone the base type profiles in sub types

1. In CentraSite Control, go to **Administration** > **Types**.

   By default, all of the available types are displayed in the **Types** tab.
2. In the **Types** tab, click the name of the sub type whose base type profiles you want to clone.
3. In the Asset Type Details page, click **Edit** to open the **Edit Asset Type** wizard.
4. On the **Edit Asset Type** wizard, click **Advanced Settings**.
5. Disable the **Inherit base type profiles** option.
6. Click **OK**.
7. In the **Clone Base Type Profiles** dialog, click **Yes** to make a copy of the base type profiles in the sub type.
8. Click **Finish**.
9. After you have cloned the base type profiles in the sub type, you can customize the cloned profiles and its attributes to suit your requirements.

Excluding Sub Types From CentraSite Business UI Search

The **Exclude sub types** option for a base type determines whether instances of its sub types should be displayed in the CentraSite Business UI. You can enable or disable this option for each base type. When you enable this option, Business UI omits instances of its sub types from the Search Results page so users cannot search for such instances. When you disable this option, Business UI includes the instances of its sub types in the Search Results page.

To exclude sub types from Business UI search

1. In CentraSite Control, go to **Administration** > **Types**.

   By default, all of the available types are displayed in the **Types** tab.
2. In the **Types** tab, click the name of the base type whose sub types you want to exclude from the search in Business UI.
3. In the Asset Type Details page, click **Edit** to open the **Edit Asset Type** wizard.
4. On the **Edit Asset Type** wizard, click **Advanced Settings**.
5. Enable the **Exclude sub types from CentraSite Business UI search** option.

6. Click **OK**, and then **Finish**.

Predefined Asset Types in CentraSite

CentraSite is installed with a number of predefined asset types. Some of these types are "core types" that belong to CentraSite itself. You can modify these types as described in "Modifications You Can Make to CentraSite's Core Asset Types" on page 266.

Other predefined types are installed to support the use of CentraSite by products such as the webMethods Product Suite. These types belong to other products, which expect the type definitions to remain unchanged. Modifying or deleting these types in CentraSite can lead to inconsistencies or errors in the product that uses the type. For example, if you modify or delete a type that is used by the webMethods Product Suite, components such as the webMethods Integration Server may no longer be able to publish assets to CentraSite. You must not modify these predefined asset types.

Predefined Asset Types Installed with CentraSite

The following table identifies the predefined asset types that are installed with CentraSite and indicates to which product they belong.

- The types that belong to CentraSite are core types that you can customize. However, you cannot delete these types.
- The types that belong to the webMethods Product Suite are custom types that are installed with CentraSite. You *must not* modify or delete these types.

Type Name	Owner
Application	CentraSite
Application Server	CentraSite
BPEL Partner	CentraSite
BPEL Partner Link	CentraSite
BPEL Partner Link Type	CentraSite
BPEL Process	CentraSite
BPEL Role	CentraSite
BPM Process Project	webMethods Product Suite

Type Name	Owner
CAF Security Role	webMethods Product Suite
CAF Task Rule	webMethods Product Suite
CAF Task Type	webMethods Product Suite
Decision Entity	webMethods Product Suite
E-form	webMethods Product Suite
Endpoint Alias	CentraSite
Event Type	webMethods Product Suite
Interface	CentraSite
IS Connection	webMethods Product Suite
IS Package	webMethods Product Suite
IS Routing Rule	webMethods Product Suite
IS Server	webMethods Product Suite
IS Service	webMethods Product Suite
IS Service Interface	webMethods Product Suite
IS Specification	webMethods Product Suite
IS Type Definition	webMethods Product Suite
JDBC Datasource	webMethods Product Suite
Operation	CentraSite
Package	CentraSite
Portlet	webMethods Product Suite

Type Name	Owner
Portlet Preference	webMethods Product Suite
Process	webMethods Product Suite
Process Pool	webMethods Product Suite
Process Step	webMethods Product Suite
Process Swimlane	webMethods Product Suite
REST Service	CentraSite
Rule Action	webMethods Product Suite
Rule Data Model	webMethods Product Suite
Rule Parameter	webMethods Product Suite
Rule Project	webMethods Product Suite
Rule Set	webMethods Product Suite
Service	CentraSite
TN Document Type	webMethods Product Suite
TN Group	webMethods Product Suite
Virtual REST Service (Virtual variant of REST Service)	CentraSite
Virtual Service (Virtual variant of Service)	CentraSite
Virtual XML Service (Virtual variant of XML Service)	CentraSite
Web Application	webMethods Product Suite
Web Application Page	webMethods Product Suite

Type Name	Owner
WS-Policy	CentraSite
XML Schema	CentraSite
XML Service	CentraSite

Modifications You Can Make to CentraSite's Core Asset Types

The following describes the ways in which you can customize the core asset types that belong to CentraSite.

- You can modify the type's existing properties and options (other than **Schema Name**, **Namespace** and **Base Type** (for Virtual variants only)).

  Note: Although CentraSiteallows you to change the display name of the predefined types, we recommend that you do not do this. Name changes might lead to problems with future upgrades of CentraSite.

- You can add custom attributes to any type other than the predefined virtual types.
- You can edit the options for existing attributes of any type other than the predefined virtual types.
- You can add profiles, delete profiles, edit profiles and rearrange the order of profiles in the type.
- You *cannot* delete any of the predefined attributes that belong to the type. You can, however, delete custom (i.e. user-defined) attributes that belong to the type.
- You *cannot* modify the inherited profiles and attributes of the predefined virtual types.

Customizing the User and Organization Types

You can add custom attributes to the User and Organization objects that are installed with CentraSite. Adding custom attributes to these types enables you to include additional metadata about the organizations or users at your site. For example, if the organizations within your enterprise belong to specific affiliates, you might want the Organization object to include an attribute that identifies the affiliate to which an organization belongs.

You can add any number of custom attributes to a User or Organization. The attributes can be of any attribute type. You can also customize the icons that are associated with these types.

When you add custom attributes to a User or Organization type, the attributes appear on the **Attributes** tab when a user or organization is displayed in CentraSite Control.

To customize the Organization or User type definition

1. In CentraSite Control, go to **Administration** > **Types**.
2. In the **Types** tab, click the Organization type or the User type, depending on which type you want to customize.
3. In the Asset Type Details page, click **Edit** to open the **Edit Asset Type** wizard.
4. In panel 1, edit the following fields if you want to use a custom icon to represent this type in the user interface.

In this field...	Do the following...
Large Icon	*Optional*. Specify the large icon that is to be used to represent this type.CentraSite Controluses this icon when it displays the details page for instances of the type. If you want to use a custom icon, click the **Browse** button and upload the file containing the large version of the icon to CentraSite. The icon you use must be in GIF format. To ensure proper alignment when the icon is displayed in the user interface, it must be 64 x 64 pixels in size.
Small Icon	*Optional*. Specify the small icon that is to be used to represent this type.CentraSite Controldisplays this icon when instances of this type appear in lists or summary tables. If you want to use a custom icon, click the **Browse** button and upload the file containing the small version of the icon to CentraSite. The icon you use must be in GIF format. To ensure proper alignment when the icons is displayed in the user interface, it must be 16 x 16 pixels in size.

5. In panel 2, edit the type's attributes as necessary. If you need procedures for this step, see "Define Attributes for a Type" on page 241.
6. Click **Finish** to save the updated type.

Deleting a Type

When you delete an asset type, keep the following points in mind:

- You can delete a type *only if there are no instances of that type in the registry*.

- The core asset types that belong to CentraSiteare non-deletable. CentraSitewill not allow you to delete these types, even if there are no instances of the selected type in the registry. For a list of the core types that belong to CentraSite, see "Predefined Asset Types Installed with CentraSite" on page 263.

Important: If you are using CentraSitein conjunction with other software products, for example, the products of the Software AGwebMethods Product Suite or a third-party product, those products can add their own asset types to CentraSite. Be aware that CentraSitetreats these types as user-defined custom types, which can be deleted by an administrator with the appropriate permissions (like any custom type). Deleting these types in CentraSitecan lead to inconsistencies or errors in the product that uses the type. For example, if you delete a type that is used by the webMethods Product Suite, components such as the webMethods Integration Server may no longer be able to publish assets to CentraSite. To prevent these types of errors, *do not delete any asset type on which other Software AGcomponents or third-party products depend*.

Deleting an Asset Type

To delete an asset type

1. In CentraSite Control, go to **Administration > Types** to display the asset types list.
2. Enable the checkbox next to the name of an asset type that you want to delete.
3. Click **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Deleting Multiple Asset Types in a Single Operation

You can delete multiple asset types in a single step. The rules described above for deleting a single asset type apply also when deleting multiple asset types.

Important: If you have selected several asset types where one or more of them are predefined types, you can use the **Delete** button to delete the types. However, as you are not allowed to delete predefined asset types, only types you have permission for will be deleted. The same applies to any other types for which you do not have the required permission.

To delete multiple asset types in a single operation

1. In CentraSite Control, go to **Administration > Types** to display the asset types list.
2. Mark the checkboxes of the asset types that you want to delete.
3. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Working with Composite Types

Certain assets can be stored in CentraSite as a set of related registry objects. Such assets are called composite assets. For example, if a web service provides several operations, this is stored in CentraSite as a composite asset consisting of the Service asset plus a separate Operation object for each of the web service's operations.

The objects that are constituents of a composite asset are referred to as components. In a composite asset there is a root component and one or more sub-components that are related to the root component. In the above example, the Service asset is the root component and the Operation objects are the sub-components. A sub-component of a composite asset can itself be a composite asset.

Depending on the relationships defined, registry operations (such as deleting an asset or exporting an asset) performed on a component of a composite asset can cause the same operation to be performed automatically on other components of the composite asset.

The concept of relationships between different objects in a SOA environment follows the UML idea of association relationships. This is only one of several forms of relationship supported by UML, but most SOA Registry Repositories only offer this form. CentraSite extends this scope to provide "aggregation" and "composition" relationships in addition to the existing association relationships. Each of these relationship forms provides its own semantics that affect specific operations that can be performed on composite assets.

You can define composite assets for all asset types, including custom (i.e. user-defined) asset types.

The CentraSite data model provides a means of representing composite assets, and allows operations to be performed on the entire composite asset or on sub-components in a consistent and well-defined manner.

The following operations take the composition definitions into account:

- Deleting an asset
- Exporting an asset
- Creating a new version of an asset
- Setting the instance permissions on an asset
- Changing the owner of an asset
- Moving an asset to another organization

Note: Lifecycle state propagation is not included in the above list, as experience has shown that such models can cause major problems in their definition and consistency rules. If such a model is required, then it should be implemented via a custom pre/post-state change policy.

Shared vs Nonshared Components

Sometimes a component can serve as a constituent of multiple composite objects. For example, XML schema "ABC" might contain schema "XYZ" as one of its components. Other services and/or schemas might also include schema XYZ as a component. Components that can belong to more than one composite object are referred to as *shared components*. Components that can only belong to a particular instance of a composite object are referred to as *nonshared components*. For example, the operations, bindings and interfaces associated with a Web service are considered *nonshared* components. These objects belong solely to the service and cannot function as constituents of other composite objects. Schemas, however, are considered *sharable*, meaning that they do not belong exclusively to a particular composite object.

Required Objects

Besides components, a composite object can also have *required objects*. Required objects are registry objects and/or repository items that are not actually part of the composite object itself, but support or augment the composite object in an essential way. For example, if a Service object has a WS-Policy attachment, the attached policy is treated as a required object because it specifies the WS-Policies that must be applied to the service when it is deployed.

Required objects, while not actually part of the composite object, must be present in the registry to make the object wholly complete or usable. (An asset's required objects are generally objects that the export process must bundle with the asset in order for the asset to be wholly represented and functional in another registry.)

Collectors

A collector is an internal process within CentraSite that identifies all of the constituents of a composite object. A collector examines a given object and returns lists that identify:

- The nonshared components associated with the composite object
- The shared components associated with the composite object
- The required objects associated with the composite object

Each composite type has its own collector. The lists produced by a collector are used by handlers that operate on instances of composite objects. For example, when you delete an XML Schema, the delete handler for schemas deletes the schema itself and all of the schema's nonshared components as identified by the collector for XML Schemas.

Defining Composite Asset Types

The relationships between components of a composite asset are defined using relationship attributes available in the appropriate asset type definition(s). A relationship can be defined on the root component or on a sub-component.

In addition to using the predefined composite asset types, you can define your own composite asset types. A user-defined composite asset type consists of the following parts:

- a user-defined asset type; each instance of this type will be the root component of a composite asset, and
- other asset types or object types; instances of these types will be related to the root component or to each other by means of relationship attributes.

The definition of a relationship may be changed at any time without affecting any instances.

You set up the associations between the components of a composite asset by using attributes of the data type "Relationship" in the asset type definition. A relationship indicates a coupling between two objects. A relationship has a direction, meaning that one of the related objects is the source of the relationship and the other object is the target of the relationship. When you define a relationship, you define it on the source object, not on the target object.

For information on how to create attributes of the data type "Relationship," see "Creating a New Type" on page 235.

The semantic of a relationship is usually indicated by the name you choose for the association type of the relationship attribute (e.g. "hasChild" or "hasParent"). You can think of the association type as a label that does not affect the behavior of the composite asset (technically it is a classification on the relationship attribute), although it makes sense to choose meaningful association types for the relationship attributes. To inform CentraSite about the semantics of the associations in your composite asset type, you need to define the relationship attributes.

CentraSite provides several forms of relationship that allow you to define plain relationships between assets as well as relationships for composite assets.

For our purposes, we will use terms and concepts introduced by UML as follows:

- **Association.** The loosest form of coupling is provided by the *association* relationship. This is like a cross-reference between two components. It indicates that there is a dependency between the components but no aggregation or composition. In this case, registry operations performed on a component do not cause any operation to be performed automatically on the related component. For example, suppose Asset A contains an association relationship to Asset B, and then Asset A is then deleted; in this case, the registry remains in a consistent state without having to delete or modify Asset B in any way.

Note: When an asset instance has an incoming relationship it may not be deleted until that incoming relationship has been removed or the asset that is the source of the relationship is in the delete set.

- **Aggregation.** A tighter coupling is provided by the *aggregation* relationship. Aggregation is similar to a whole/part relationship in which components of a structure can also exist independently of the structure; this is like the "contains" semantic, whereby one component contains another component but does not own it. In this case, some operations performed on a component cause the same operation to be performed automatically on the related component. For example, if you want to export an asset, CentraSite automatically extends the export set by adding all of the components that are coupled by aggregation. However, if you want to delete an asset, CentraSite leaves the coupled components unchanged.
- **Composition.** The tightest coupling is provided by the *composition* relationship. Composition is similar to a whole/part relationship in which components of a structure cannot exist independently of the structure; this is like the "owns" semantic, whereby one component owns another component. In this case, all registry operations performed on a component cause the same operation to be performed automatically on the related components. For example, if you want to delete an asset, then CentraSite automatically extends the delete set by adding all of the components that are coupled by composition.

The form of relationship determines the way in which registry operations performed on one component affect the related components. In the following table, entries marked with "yes" mean that an operation on a component causes the same operation to be performed on the related components, whereas table entries marked with "no" mean that the related components are not changed.

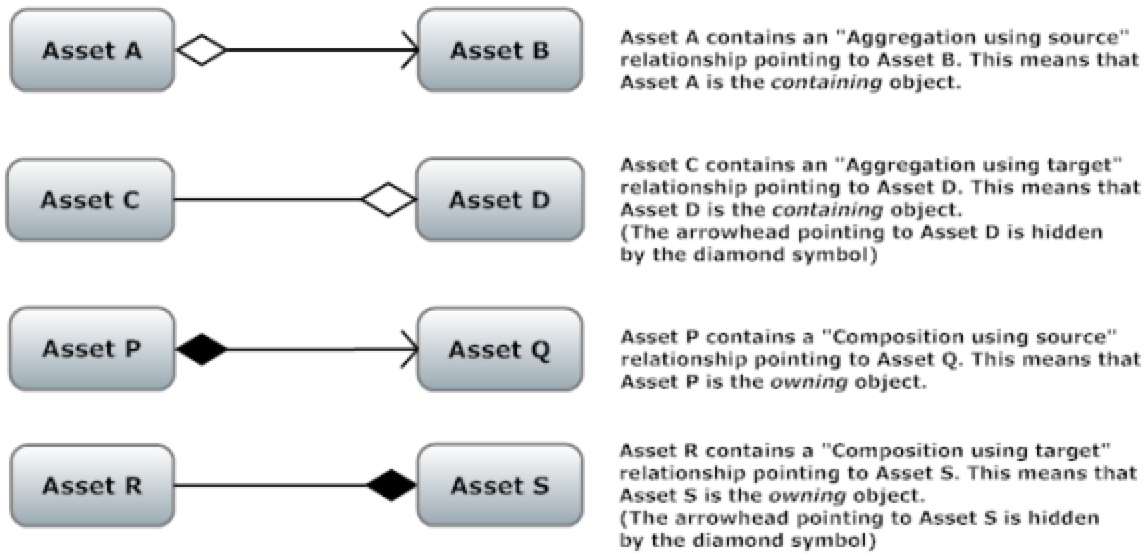
Operation on component	Form of Relationship:		
	Association	Aggregation	Composition
Move asset to another organization	no	no	yes
Change asset owner	no	no	yes
Delete asset	no	no	yes
Export asset	no	yes	yes
Set instance permissions on an asset	no	yes	yes
Create new version of an asset	no	no	yes

These operations are the primary set which are affected by different forms of relationships and are supported out-of-the-box by CentraSite.

Aggregation and Composition come in two forms, namely "with source" and "with target":

- **Aggregation/Composition with source.** This means that the aggregation or composition treats the source component (i.e. the component where the relationship is defined) as the containing or owning component, and the target component (i.e. the component that the relationship points to) as the contained or owned component.
- **Aggregation/Composition with target.** This means that the aggregation or composition treats the source component (i.e. the component where the relationship is defined) as the contained or owned component, and the target component (i.e. the component that the relationship points to) as the containing or owning component.

You might find the following diagrams useful to illustrate the relationships in composite assets. They are similar to UML diagrams, but allow the aggregation or composition to be on the target component (in UML, they can only be on the source component). The forms "with source" and "with target" are represented using a diamond-shaped symbol to indicate the containing/owning component. Aggregation is indicated by a non-filled diamond symbol and Composition is indicated by a filled diamond symbol. An arrow points from the source component to the target component, with the arrowhead located at the target component. If the diamond symbol and the arrowhead are located at the same component, only the diamond is shown.



Asset A contains an "Aggregation using source" relationship pointing to Asset B. This means that Asset A is the *containing* object.

Asset C contains an "Aggregation using target" relationship pointing to Asset D. This means that Asset D is the *containing* object. (The arrowhead pointing to Asset D is hidden by the diamond symbol)

Asset P contains a "Composition using source" relationship pointing to Asset Q. This means that Asset P is the *owning* object.

Asset R contains a "Composition using target" relationship pointing to Asset S. This means that Asset S is the *owning* object. (The arrowhead pointing to Asset S is hidden by the diamond symbol)

Semantics of Relationships and Operations

Association Relationship

Association relationships are the relationships that were available in later releases of CentraSite v8 and are available with unchanged semantics in the current release.

Aggregation Relationship

The aggregation relationship changes the rules in the following way for operations:

Operation	Rules
Delete an asset	There are no changes in the delete rules when introducing aggregation.
Create a new version of an asset	There are no changes in the versioning rules when introducing aggregation. However, for assets of type "Service" and "XML Schema", there is an additional possibility: If you mark the checkbox "Propagate to dependent objects" when you create a new version of the root component of a composite asset of one of these types, the versioning will be propagated also to components of these types that are connected to the root component via aggregation relationships.
Set instance permissions on an asset	Merges the permissions of the initiating component with those of the current component. The permissions assigned to the contained component are the union of the permissions of the containing asset and the contained component. If the user that performs the operation does not have Full permissions on a component, then it and all of its sub-components will be skipped.
Move asset to another organization	There are no changes in the move organization rules when introducing aggregation.
Change the owner of an asset	There are no changes in the change owner rules when introducing aggregation.
Export an asset	If a component that has a *containing* aggregation is added to the export set, then the target is also added to the export set. The rules when selecting the checkbox 'including instances' in the user interface apply as before with the addition of the containing rules.

Note: For Export, the usage of recursive relationships on the type and instance level must be taken into account. Whereby type level does not mean that the same instance is referenced.

Composition Relationship

Composition relationships affect all of the defined operations to varying degrees.

Operation: Deleting an Asset

On deletion, if the root component is added to the set to be deleted, then the sub-components will also be added to the set to be deleted. The direction of the association does not play a role in defining the set, only the *containing* designation. This means it is possible for the deletion to fail if one of the assets added to the deletion set during this processing is referenced via the basic association relationship target rules.

This rule is applied recursively. For example, if we have three assets that have the relationships "A contains B contains C", then the following statements apply:

- When A is deleted, then B will be deleted and finally because B is deleted, C will also be deleted.
- When deleting C, only C will be deleted.

This fails if the deleting user does not have permission on any of the assets in the set acquired by traversing the graph. The delete is considered atomic - either all are deleted or none. This avoids inconsistencies in the outcome of the operation.

The relationship direction always plays a role in the deletion operation. An asset may not be deleted if it is the target of a relationship and the source is not part of the deletion set.

The deletion rules described here apply also when you purge old versions of an asset. In this case, the purge operation will be applied not only to the component being purged, but also to the related sub-components.

Operation: Creating a New Version of an Asset

On versioning, if the root component is added to the set to be versioned, then the sub-components will also be added to the set. The direction of the association does not play a role in defining the set, only the *containing* designation.

If you create a new version of an asset that is the root component of a composite asset, and the root component is related to one or more of the other components via composition relationships, CentraSite automatically creates a new version of each of these other components.

Operation: Exporting an Asset

On export, if the root component is added to the set to be exported, then the sub-components will also be added to the set. The direction of the association does not play a role in defining the set, only the *containing* designation.

Operation: Setting Instance Permissions on an Asset

On setting permissions, when the root component is added to the set to which the permissions will be applied, then the sub-components asset are also added if and only if the user has the permission to modify the permission of the target. If the user does not have permission, then the graph traversal for the target is not carried further for this sub-graph.

The permission set that will be given to all sub-component assets is the merge based on what is to be modified. The permissions assigned to the owned asset are the union of the permissions of the owning asset and the owned asset.

Operation: Moving an Asset to Another Organization

On moving an asset to another organization, when the root component is added to the set to be moved, then the sub-components are also added. No permission checks are done during this operation as only users in the CentraSite Administrator role may perform this operation.

Operation: Changing the Owner of an Asset

On changing ownership, when the root component is added to the set to be changed, then the sub-components are also added to the set. No permission checks are done during this operation as only users in the CentraSite Administrator role may perform this operation.

Extended Rules

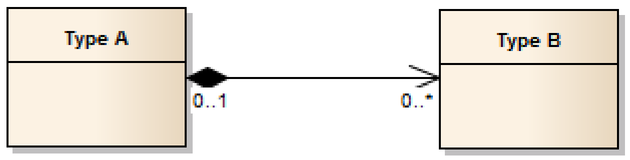
Changing Relationships

As part of the support for Aggregation and Composition, CentraSite allows the relationship form to be changed after the type is created. This change affects all current instances and new instances. This means that after a relationship attribute is created, the form (for association the default form, for aggregation (both forms, i.e. "using source" and "using target") and for composite (both forms)) can be changed by an Asset Type Administrator. From that point onwards, the appropriate rules will be applied when performing the defined operations.

Updating Assets

The following asset updates need to be taken into account when implementing models:

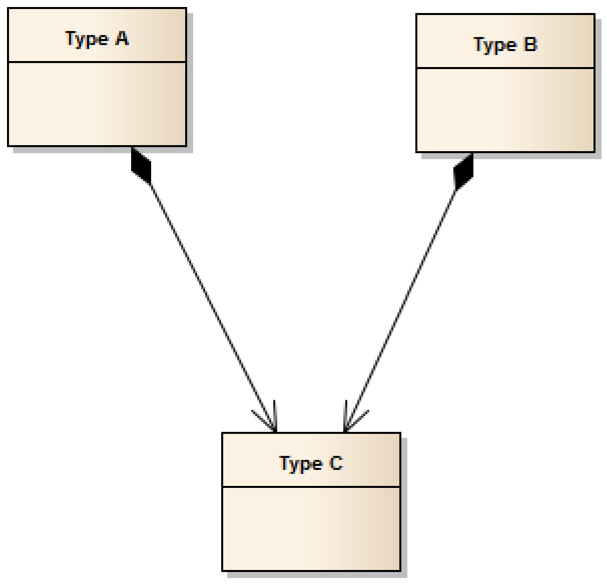
1. Adding relationships to existing instances. Given the below model:



   It is perfectly legal to create instances of Type A and Type B independent of one another. In fact in CentraSite this characteristic is mandatory, as the creation of multiple assets at the same time is only allowed in a few places in the UI.

When adding the relationship between an instance of Type A and an instance of Type B, CentraSite will not do any extra operations to guarantee the consistency of permissions at this point.

2. Adding relationships to 2 different composites. Given the below model (which is a legal model):



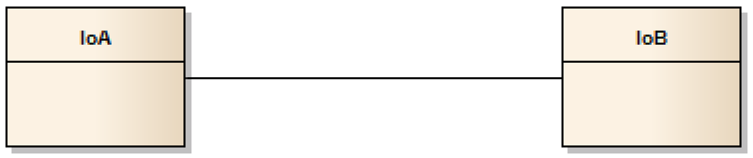
The following restriction applies to user-defined asset types, but not predefined asset types: At runtime if an instance of Type A creates a composite relationship to an instance of Type C, and then an instance of Type B tries to create a composite relationship to the same instance of Type C, this composition will be rejected. This is because a contained asset (instance of Type C) can only have one owning asset (instance of Type A or instance of Type B).

Usage Scenarios

Overview

The outcome of each operation is given based on a very simple type and instance configuration in each of the usage scenarios.

Unless otherwise stated, the instances that each operation will be performed on will be:

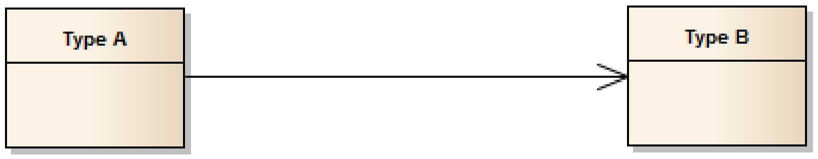


Key	Description
IoA	Instance of type A
IoB	Instance of type B

Delete Usage Scenarios

Association Relationship

Given the type model:

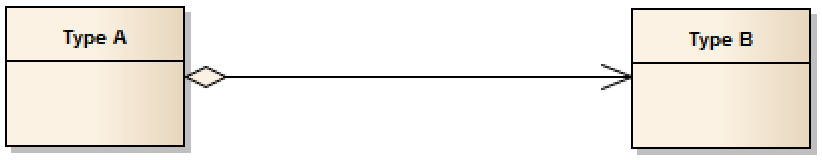


The result of the delete operation will be:

Operation	Expected result
Delete IoB	Fail because of incoming relationship from IoA
Delete IoA	Success. Post-condition: IoB will be left intact

Aggregation Relationship with Containing Constraint on Type A

Given the type model:

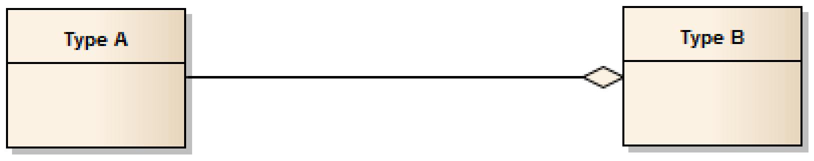


The result of the delete operation will be:

Operation	Expected result
Delete IoB	Fail because of incoming relationship from IoA
Delete IoA	Success. Post-condition: IoB will be left intact

Aggregation Relationship with Containing Constraint on Type B

Given the type model:

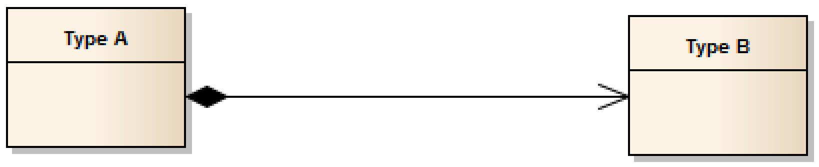


The result of the delete operation will be:

Operation	Expected result
Delete IoB	Fail because of incoming relationship from IoA
Delete IoA	Success. Post-condition: IoB will be left intact

Composition Relationship with Containing Constraint on Type A

Given the type model:

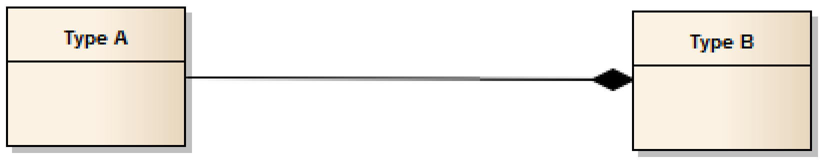


The result of the delete operation will be:

Operation	Expected result
Delete IoB	Fail because of incoming relationship from IoA
Delete IoA	Success. Post-condition: IoB will be removed

Composition Relationship with Containing Constraint on Type B

Given the type model:

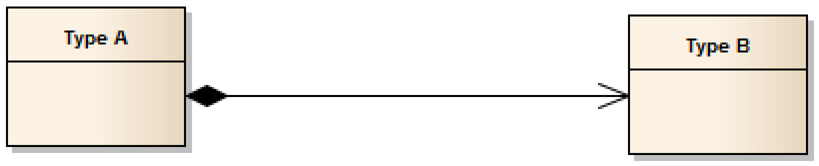


The result of the delete operation will be:

Operation	Expected result
Delete IoB	Success. Post-condition: IoA will be removed
Delete IoA	Success. Post-condition: IoB will be left intact

Composition Relationship with Permission Scenario

Given the type model:



With the constraints:

- User who will perform the deletion is Fred
- Fred has Full permission on IoA
- Fred has Read permission on IoB

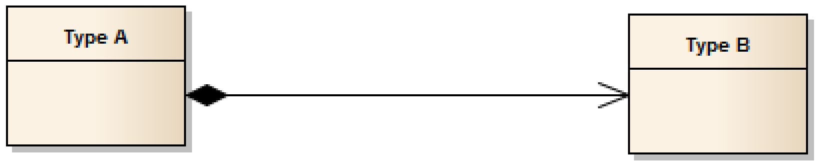
The result of the delete operation will be:

Operation	Expected result
Delete IoB	Fail. User Fred does not have full permission on IoB
Delete IoA	Fail. User Fred does not have full permission on IoB

Versioning Usage Scenarios

Versioning for Association Relationship and Aggregation Relationship are the same and do not change from previous versions, therefore only Composition Relationship are shown below.

Given the type model:

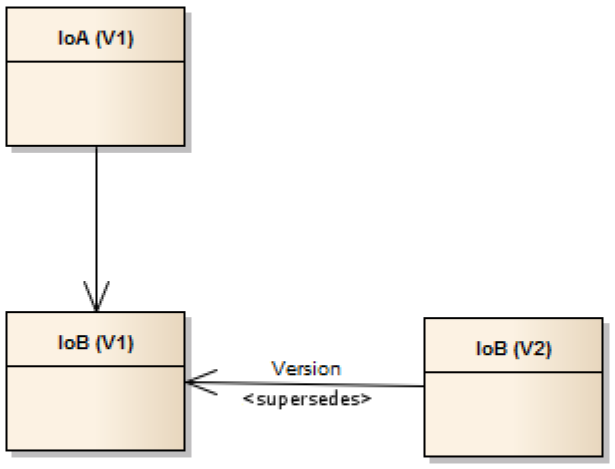


Note: Both variants of a composite relationship (source and target) are supported and are orthogonal.

Based on the instances given above, the following scenarios are considered relevant.

Versioning of IoB

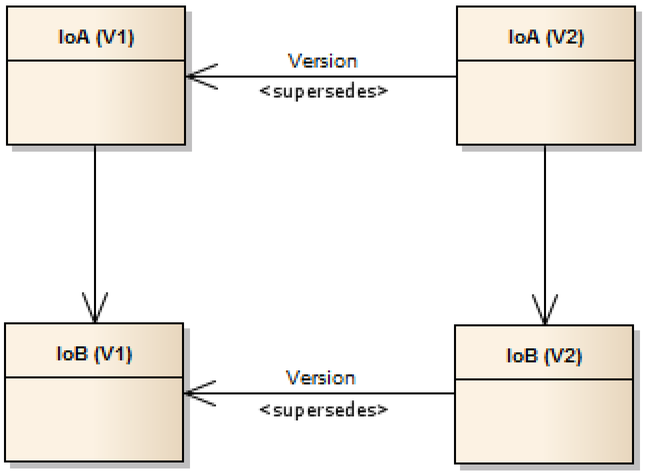
This causes just IoB to be versioned, and the IoA version is left unchanged. Pictorially, this looks like:



Versioning of IoA

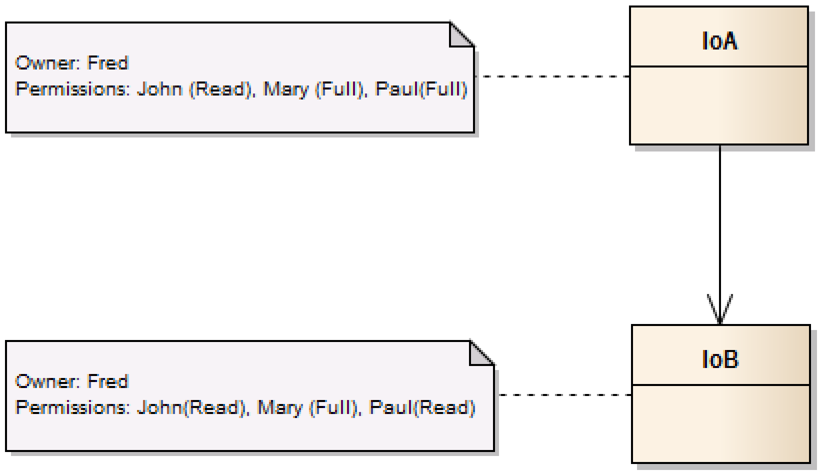
Versioning of IoA will result in the composite relationship being used to work out which other assets should be versioned at the same time. This will result in IoA and IoB being versioned together (if we fail to version either then neither will be versioned).

This pictorially looks like:



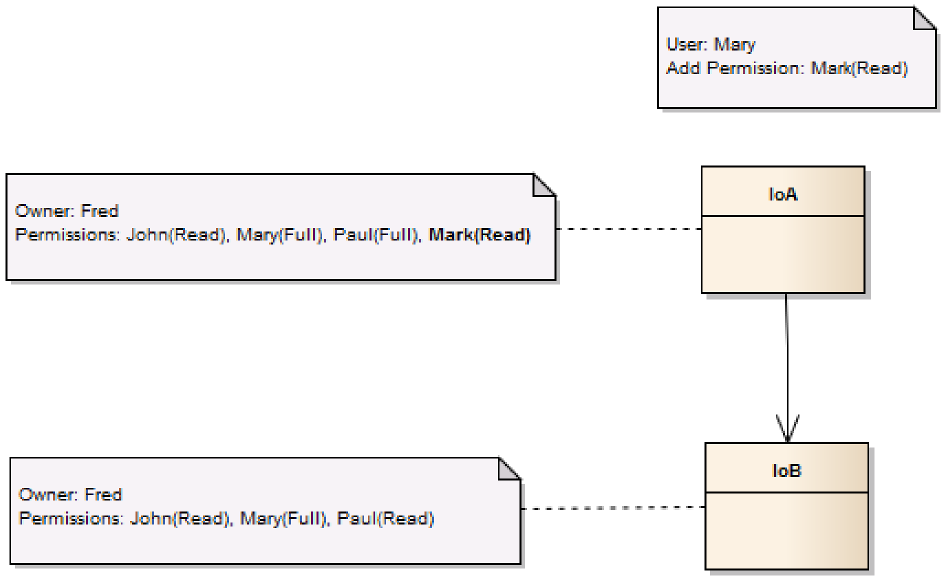
Permission Usage Scenarios

For the permission scenarios, the following instances with the annotations for the owner and permissions will be used as basis.


Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full)
IoA
IoB
Owner: Fred
Permissions: John(Read), Mary (Full), Paul(Read)

Association Relationship Permission Propagation

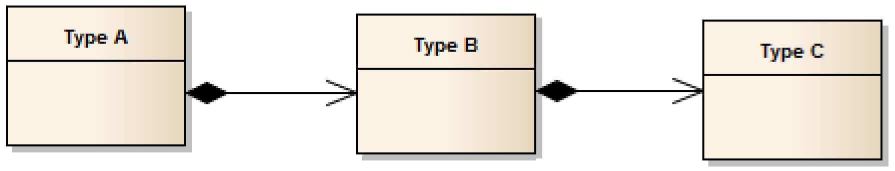
When an Association Relationship is used, the permission propagation does not take place. This means that if an instance permission is set, then only that instance's permission is affected. This means that if user Mary adds Read permission for user Mark on IoA, then Mark only gets permission to Read IoA. He does not get permission to Read IoB. Pictorially this looks like:


User: Mary
Add Permission: Mark(Read)
Owner: Fred
Permissions: John(Read), Mary(Full), Paul(Full), **Mark(Read)**
IoA
IoB
Owner: Fred
Permissions: John(Read), Mary(Full), Paul(Read)

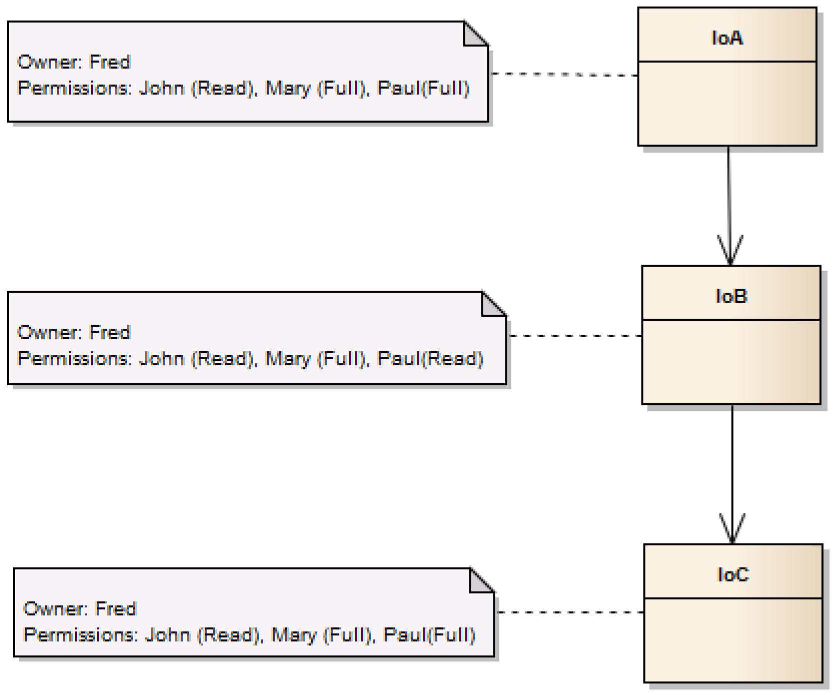
Composition/Aggregation with Weak Propagation

One of the key points of permission propagation is what happens if a user only has a Full permission on a subset of the assets to which the permissions need to be propagated. In this case, the usage of the so-called weak propagation rule comes into effect. The rule states that if a user does not have permission to propagate to all instances in the set, then the permissions will be propagated to only the instances which are allowed.

For this scenario the following model will be used:

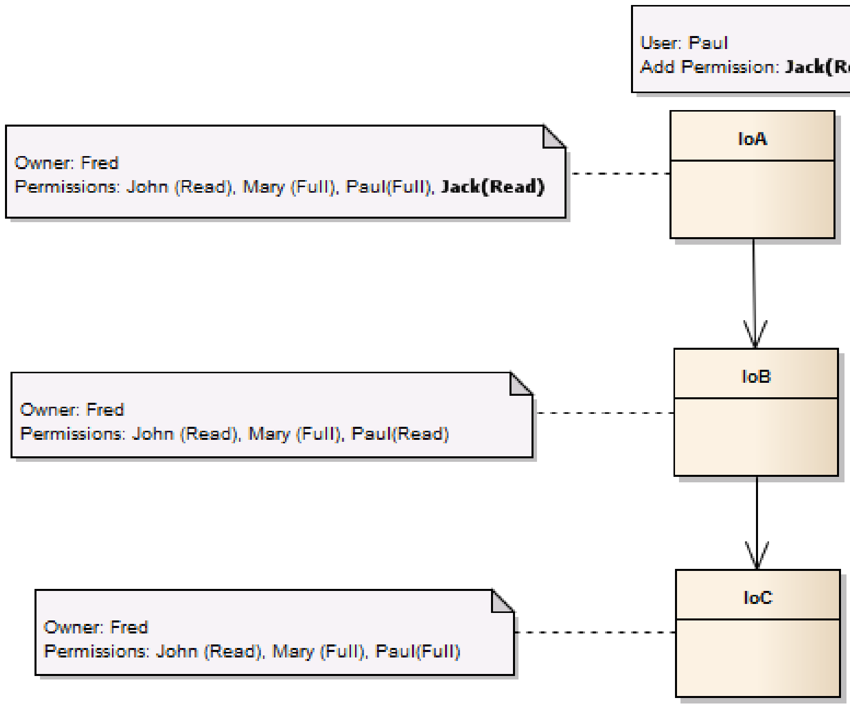


Based on this model, the instances and the permissions to start with should be:


Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full)
IoA
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Read)
IoB
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full)
IoC

Now if user Paul adds Read permission for user Jack to IoA, Jack will only get this permission on IoA as Paul does not have the rights to give Jack the permissions on IoB. Even though Paul has Full permission on IoC, because it is a child of IoB, Jack does not get the permissions for IoC because of weak propagation. We trim/terminate the propagation at IoB.

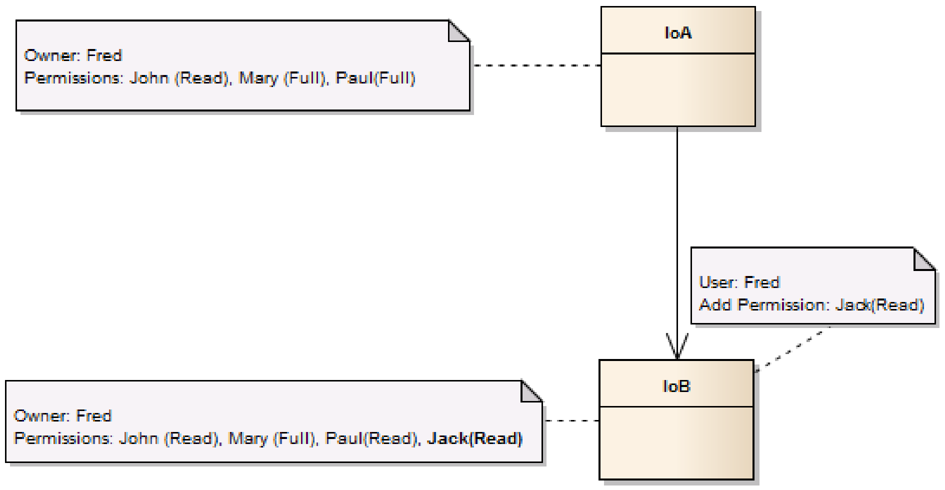
This pictorially looks like:


User: Paul
Add Permission: Jack(Read)
IoA
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full), Jack(Read)
IoB
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Read)
IoC
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full)

Composition/Aggregation Relationship Updating Sub-component

Updating the permissions of a sub-component without affecting the overall composition/aggregation is not affected with the changes. Therefore if user Fred wants to explicitly add Read permission for Jack on IoB, this is possible.

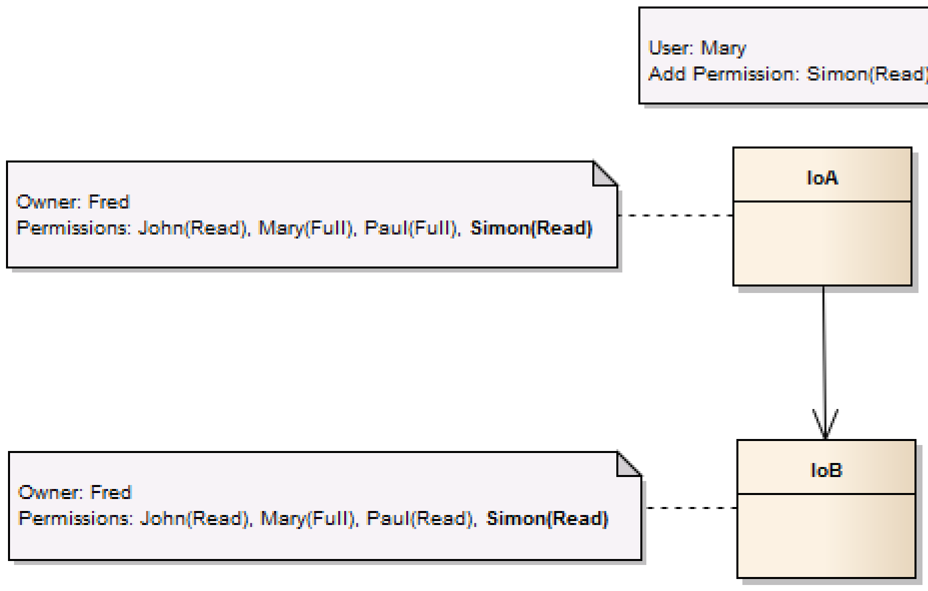
This pictorially looks like:


IoA
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Full)
User: Fred
Add Permission: Jack(Read)
IoB
Owner: Fred
Permissions: John (Read), Mary (Full), Paul(Read), Jack(Read)

Composition/Aggregation Relationship with Full Propagation

Full propagation happens when all sub-components can be updated by the instigating user. As example, Mary wants to give Read permission to Simon on IoA with permission propagation via the Composition/Aggregation relationship. As Mary has Full permissions on IoA and IoB, the permissions are propagated over the relationship.

This pictorially results in:

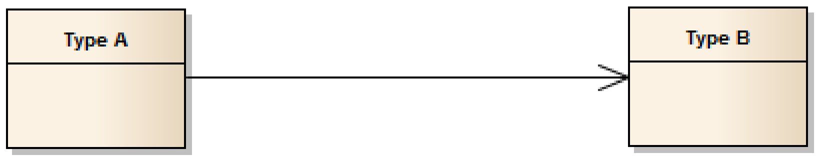

User: Mary
Add Permission: Simon(Read)
Owner: Fred
Permissions: John(Read), Mary(Full), Paul(Full), **Simon(Read)**
IoA
Owner: Fred
Permissions: John(Read), Mary(Full), Paul(Read), **Simon(Read)**
IoB

Export Usage Scenarios

Export with Association Relationship

Export will work the same as in previous versions - the usage given here assumes that no additional options are chosen.

Given the model:



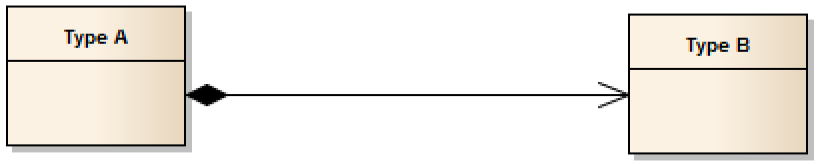
The result of the Export operation will be:

Operation	Expected result
Export IoB	Export set contains IoB. It does not contain IoA
Export IoA	Export set contains IoA. It does not contain IoB

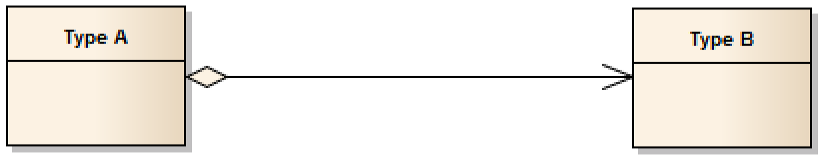
Export with Composition/Aggregation Relationship

For both Composition and Aggregation Relationships, the rules are exactly the same. When such a Relationship with the appropriate containing rule is found, then traverse the relationship and add sub-components to the set.

Given the model:



OR



The result of the Export operation will be:

Operation	Expected result
Export IoB	Export set contains IoB. It does not contain IoA
Export IoA	Export set contains IoA and IoB.

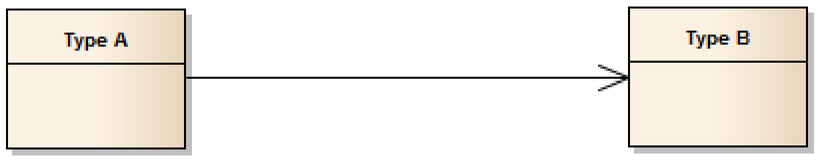
Move Organization Usage Scenarios

Move organization is an administrative task and may only be performed by someone with appropriate administration rights. This means that permissions and ownership do not play a role when performing the move operation.

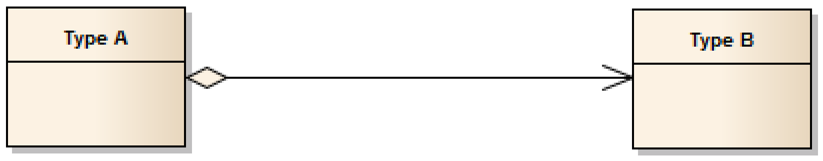
Move Organization with Association/Aggregation Relationship

When moving an asset from one organization to another, the Association and Aggregation Relationships do not change any related assets. This is because both of these relationships are considered to be loosely coupled.

Given the model:



OR



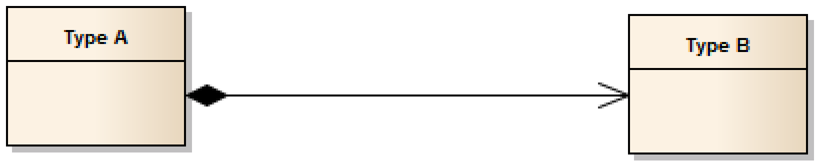
The result of the Move Organization operation will be:

Operation	Expected result
Move IoB	Only IoB will be moved to the new organization. It does not move IoA.
Move IoA	Only IoA will be moved to the new organization. It does not move IoB.

Move Organization with Composition Relationship

When a Composition Relationship with the appropriate containing rule is found, then traverse the relationship and move the sub-components to the new organization.

Given the model:



The result of the Move Organization operation will be:

Operation	Expected result
Move IoB	Only IoB will be moved to the new organization. It does not move IoA.
Move IoA	IoA and IoB will be moved to the new organization.

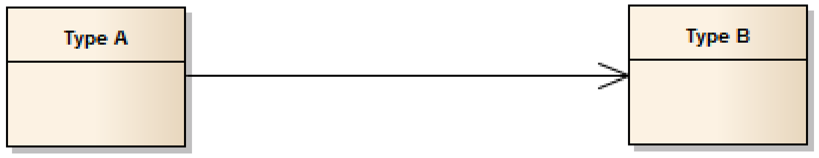
Change Ownership Usage Scenarios

Change Ownership is an administrative task and may only be performed by someone with appropriate administration rights. This means that permissions and ownership do not play a role when performing the change ownership operation.

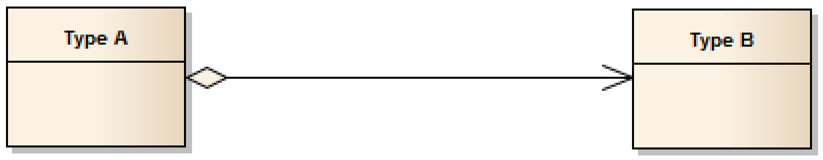
Change Ownership with Association/Aggregation Relationship

When changing an asset's ownership from one user to another, the Association and Aggregation Relationships do not change any related assets. This is because both of these relationships are considered to be loosely coupled.

Given the model:



OR



The result of the Change Ownership operation will be:

Operation	Expected result
Change Ownership of IoB	Change the ownership of IoB. It does not affect IoA.
Change Ownership of IoA	Change the ownership of IoA. It does not affect IoB.

Propagation of Profile Permissions

In addition to propagating permissions that control the access to an asset instance (as described above), it is also possible to propagate permissions that control the access to the asset instance's profiles.

Profile permissions of the root asset of a composite asset can be propagated to the other components if the components have the same type as the root asset. This restriction arises because different asset types can have different sets of profiles, whereas assets of the same type have the same set of profiles.

Propagation of profile permissions is activated when you mark the checkbox **Propagate profile permissions** in the asset's **Permissions** tab. This checkbox can only be selected if you have also marked the checkbox **Propagate Permissions to dependent objects**.

Predefined Composite Asset Types

The following sections identify the nonshared components, shared components, and required objects that are associated with each of the predefined composite types installed with CentraSite.

Service

Nonshared Components	Description
Binding(s)	The objects that represent the specific ports that are defined in the WSDL. (A port defines a specific endpoint where the service is provided.)
Interface(s)	The objects that represent the portType that is a defined in the WSDL. (A portType defines a set of operations that the service provides.)
Operation(s)	The objects that represent the individual operations that the service provides.
Service WSDL	The ExternalLink to the WSDL file and the WSDL file itself.
Other WSDL (references to WSDLs that are imported or included in the main service WSDL)	The ExternalLinks to the referenced WSDL files and the referenced WSDL files themselves. **Important:** If, at collection time, the collector discovers that a referenced WSDL is also a component of another asset, it places that WSDL in the *shared component* list. Otherwise, it returns the referenced WSDL in the list of nonshared components.
BPEL Partner Link Type	The object that represents the BPEL Partner Link Type to which the service is related (if such an association exists).
BPEL Role	The object that represents the BPEL Role to which the service is related (if such an association exists).

Shared Components	Description
XML Schema(s)	The entire graph of XML schemas that are related to the service. (The graph includes all of the XML schemas that the service references directly or indirectly). For each XML schema in the graph, the collector collects the ExternalLink to the XSD file and the actual XSD file itself.

Required Objects	Description
Service type	The Type object that defines the structure of a Service object in this registry (including all user-defined profiles that have been defined for the type).
WS-Policy	The WS-Policy objects that are associated with the service (if any).
Supporting Documents	ExternalLinks that point to files in the supporting document library plus the files themselves (if any). **Note:** The list of supporting documents that the collector returns includes: ■ documents that have been attached to the service using any of the predefined File attributes defined in the Service type ■ documents that have been attached to the service using a custom File attribute ■ any documents that have been attached to the service using an ad-hoc ExternalLink

Virtual Service

Nonshared Components	Description
The set of nonshared components defined for the Service type.	See nonshared components in "Service" on page 290.
WS-Policy	The WS-Policy object associated with the virtual service.

Nonshared Components	Description
Processing Steps	The policy objects that represent the processing steps for the virtual service.
Extrinsic object	An internal copy of the WSDL that is maintained for the virtual service.
VSD	The virtual service's virtual service descriptor (VSD).

Shared Components	Description
The set of shared components defined for the Service type.	See shared components in "Service" on page 290.

Required Objects	Description
Service type	The Type object that defines the structure of a Service object in this registry (including all user-defined profiles that have been defined for the type).
Native service	The Service object from which the virtual service was generated.
Supporting Documents	ExternalLinks that point to files in the supporting document library plus the files themselves. **Note:** The list of supporting documents that the collector returns includes: ■ documents that have been attached to the service using any of the predefined File attributes defined in the Service type ■ documents that have been attached to the virtual service using a custom File attribute ■ any documents that have been attached to the virtual service using an ad-hoc ExternalLink

XML/REST Service

Nonshared Components	Description
ServiceBinding(s)	Each ServiceBinding object represents an <endpoint/> element of the XML/REST Service WSDL20.
Binding Concept(s)	Each Binding Concept object represents a <binding/> element of the XML/REST Service WSDL20.
Interface(s)	Each Interface object represents an <interface/> element of the XML/REST Service WSDL20.
SpecificationLink(s)	SpecificationLinks are used to link the Interface and Binding Concept objects to a ServiceBinding object.
Operation(s)	Each Operation object represents an XML/REST Service Resource and is represented as an element in the WSDL20.
Service WSDL20	The ExternalLink to the WSDL20 file (in the CentraSite repository) and the WSDL20 file itself.

Shared Components	Description
XML Schema(s)	The entire graph of XML Schemas that are related to the XML/REST Service. (The graph includes all of the XML Schemas that the XML/REST Service's Resources reference directly or indirectly). For each XML Schema in the graph, the collector collects the ExternalLink to the XSD file (in the CentraSite repository) and the actual XSD file itself.

Required Objects	Description
Service Type	The ObjectType Concept that defines the structure of a Service object in this registry (including all user defined profiles that have been defined for the type).
CentraSiteVirtualType	The CentraSiteVirtualType Concept that identifies the VirtualType of the Service object.

Required Objects	Description
WS-Policy	The WS-Policy objects that are associated with the service (if any).
Supporting Documents	ExternalLinks that point to files in the supporting document library plus the files themselves (if any). **Note:** The list of supporting documents that the collector returns includes: ■ documents that have been attached to the service using any of the predefined File attributes defined in the Service type ■ documents that have been attached to the service using a custom File attribute ■ any documents that have been attached to the service using an ad-hoc ExternalLink

Virtual XML/REST Service

Nonshared Components	Description
The set of nonshared components defined for the XML/REST Service type.	See nonshared components under "XML/REST Service" on page 293.
WS-Policy	The WS-Policy object associated with the virtual service.
Processing Steps	The policy objects that represent the processing steps for the Virtual XML/REST Service.
Extrinsic object	An internal copy of the WSDL20 that is maintained for the Virtual XML/REST Service.
VSD	The Virtual XML/REST Service's virtual service descriptor (VSD).

Shared Components	Description
The set of shared components defined for	See shared components under "XML/REST Service" on page 293.

Shared Components	Description
the XML/REST Service type.	

Required Objects	Description
Service type	The ObjectType Concept that defines the structure of a Service object in this registry (including all user-defined profiles that have been defined for the type).
CentraSiteVirtualType	The CentraSiteVirtualType Concept that identifies the VirtualType of the Service object.
Native service	The XML/REST Service object from which the Virtual XML/REST Service was generated.
Supporting Documents	ExternalLinks that point to files in the supporting document library plus the files themselves. **Note:** The list of supporting documents that the collector returns includes: ■ documents that have been attached to the service using any of the predefined File attributes defined in the Service type ■ documents that have been attached to the virtual service using a custom File attribute ■ any documents that have been attached to the virtual service using an ad-hoc ExternalLink

XML Schema

Nonshared Components	Description
XSD File	The ExternalLink to the XSD file and the XSD file itself.
XML Schema(s) (referenced)	The entire graph of XML schemas that are related to this XML schema. (The graph includes all of the schemas that this XML schema references directly or indirectly). For each XML schema in the graph, the collector collects the ExternalLink to the XSD file and the actual XSD file itself.

Shared Objects	Description
None	n/a

Required Objects	Description
XML Schema type	The Type object that defines the structure of an XML Schema object in this registry (including all user-defined profiles that have been defined for the type).

BPEL Process

Nonshared Components	Description
BPEL File	The ExternalLink to the BPEL document and the BPEL document itself.
BPEL Partners	The objects that represent partners in the BPEL process.
PartnerLinks	The objects that represent partner links in the BPEL process.
Association Type (Service)	The association type that CentraSite uses to relate a BPEL process to a service.

Shared Objects	Description
None	n/a

Required Objects	Description
BPEL type	The Type object that defines the structure of a BPEL object in this registry (including all user-defined profiles that have been defined for the type).
Services	The services that are referenced by the BPEL's PartnerLinks. (This includes all of the components and required objects associated with the referenced

Required Objects	Description
	services. For a detailed list of these components and required objects, see "Service" on page 290.)
PartnerLinkTypes	The PartnerLinkTypes that are referenced by the BPEL's PartnerLinks.
Roles	The Roles that are referenced by the BPEL's PartnerLinks.

WS-Policy

Nonshared Components	Description
Policy	The object representing the policy itself.
Policy Parameter	The objects which form the parameters defined for the policy (if any). **Note:** There can be multiple levels of parameters which can be nested; parameters from all levels will be included.
Policy Condition (if any)	The object representing the conditions defined for the policy (for example, if the policy has conditions such as "Name contains XML").

Shared Objects	Description
None	n/a

Required Objects	Description
Custom policy action	A custom action that is used by the policy and is not available as part of the predefined set.
Custom policy action parameters	All action parameters used by a custom action that the policy uses (if any).
Custom asset types	The custom defined asset types that the policy is defined for (if any).

Required Objects	Description
Registry object	The registry object that the policy uses as a parameter (if any).

BPM Process Project

Components	Description
Process	Contains a set of Process Steps that invokes services and possibly other processes and has documents as inputs and outputs.

BPM Process Step

Components	Description
CAF Task Type	webMethods Task Engine human activity task.
CAF Security Role	A role which has the privilege to participate in CAF actions.
E-Form	The E-form object represents electronic forms that support forms-driven processes.
IS Service	Represents a webMethods IS Service Type.
Process Pool	Allows grouping of process steps into an internal or external process. More than one pool may exist within a process.
Process	Contains a set of Process Steps that invokes services and possibly other processes and has documents as inputs and outputs.
Service	A service is a software component that is described via a well defined interface and is capable of being accessed via standard network protocols such as, but not limited to, SOAP over HTTP. CentraSite is able to extract metadata of services based on a WSDL description.

Components	Description
User	The predefined JAXR-based type User.
XML Schema	A reference to an XML Schema file

IS Package

Components	Description
IS Specification	Represents a webMethods IS Specification Type.
IS Type Definition	Represents a webMethods IS Type Definition Type.
IS Routing Rule	Represents a webMethods IS Routing Rule Type.
IS Service Interface	Represents a webMethods IS Service Interface Type.
IS Connection	Represents a webMethods IS Connection Type.
IS Service	Represents a webMethods IS Service Type.

IS Service Interface

Components	Description
Binding(s)	The objects that represent the specific ports that are defined in the WSDL. (A port defines a specific endpoint where the service is provided.)
Interface(s)	The objects that represent the portType that is a defined in the WSDL. (A portType defines a set of operations that the service provides.)
Operation(s)	The objects that represent the individual operations that the service provides.
IS Service	Represents an Integration Server (IS) Service operation.

Components	Description
REST Service	Represents a corresponding REST Service in the Integration Server (IS).

Process

Components	Description
Process Step	Represents an activity in a Process.
Process Pool	Allows grouping of Process Steps into an internal or external Process. More than one Process Pool may exist within a Process.

Required Object	Description
BPM Process Project	A user defined project that allows users to group BPM assets.

Process Pool

Components	Description
Process Swimlane	Allows the grouping of Process Steps by actor.

Process Swimlane

Components	Description
Process Step	Represents an activity in a Process.

Web Application

Components	Description
Web Application Page	Construct that is used to build User Interface pages for CAF web, portlet and task applications.

Components	Description
Portlet	Portlet built using webMethods CAF User Interface technology. Supports JSR 168.
CAF Task Type	webMethods Task Engine human activity task.
CAF Security Role	Java Web Application Security Role.
JDBC Datasource	Connection to a JDBC database.

Portlet

Components	Description
Portlet Preference	Allows customized portlet behavior.
Web Application Page	Construct that is used to build User Interface pages for CAF web, portlet and task applications.

CAF Task Type

Components	Description
CAF Task Rule	webMethods Task Engine Task Assignment or Event execution mechanism.

TN Group

Components	Description
TN Document Type	A description of a document type that is expected in a user's Trading Network.

Business Rules Project

Components	Description
Rule Set	A container for related rule metaphor assets.

Components	Description
Data Model	Defines a set of data elements available to a rule.
Rule Action	Represents some external behavior.

Data Model

Components	Description
Rule Parameter	Represents the connection from a metaphor to a data model.

Working with Association Types

An Association Type is a logical definition of a relationship. An association type enables you to define a relationship between an asset of one type and an asset of another type and/or to any other object defined in CentraSite. The association type represents the relationship as a forward label or reverse label. An association type can be used by the relationship attribute defined within an asset's profile. For example, if you have an **Association Type** called `Uses`, you can define a **Relationship** attribute with this association type `Uses` for a Service asset. You can use that attribute to link the Service asset to other Services in the catalog, to assets of other types (such as XML schemas) and/or to other registry objects (such as Organizations, Policies). For an **Asset Type**, when you define a relationship attribute using this **Association Type** it appears as a attribute on the asset's detail page in the CentraSite Control.

One association type can be used by many asset types. For example, the `Has Parent` association type, which is one of the predefined association types installed with CentraSite, provides parent-child relationship information about an asset and the related to object, and is used by both the Service asset type and the XML schema asset type.

Who Can Create and Manage Association Types?

To create custom association types, you must belong to a role that has the Manage Asset Types permission. Besides allowing you to create custom association types, this permission allows you to edit and delete any user-defined association type. Additionally, it allows you to edit certain predefined association types installed by CentraSite. By default, users in the CentraSite Administrator and Asset Type Administrator roles have this permission, although an administrator can grant this permission to other roles.

For more information about permissions, see "About Roles and Permissions" on page 115.

Adding an Association Type

You use the following procedure to define a new association type.

To define a new association type

1. In CentraSite Control, go to **Administration > Types**. Click the **Association Types** tab.
2. Click **Add Association Type**.
3. In the **Add Association Type** dialog box, specify the following properties:

In this field...	Do the following...
Name	Enter a name for the association type. Be aware that this is the name that will be given to attributes that use this association type. Therefore, the name should be meaningful when used as an attribute name. For example, use an association name such as `Developed By`, not `developer association`. ■ An association name does not need to be unique within the CentraSite registry. However, to reduce ambiguity, you should avoid giving multiple associations the same name. ■ An association name can contain any character (including spaces).
Forward Label	Specify the relationship of the source asset (the one in which the `Relationship` attribute resides) to one or more specified targets. If you are not specifying a name for the forward label, then CentraSite will treat the association type name as the forward label.
Reverse Label	*Optional*. Specify the relationship of the specified targets to the source asset.

4. When you finish setting the association type's properties, click **OK**.

Editing the Properties of an Association Type

When you attempt to modify an association type, keep the following points in mind:

- You can modify the association type's name at any time.
- You can modify an association type for relationship properties *only if there is no relationship attribute defined with it in the catalog*. After a association type has been assigned to an attribute, it can no longer be edited.

To modify an association type

1. In CentraSite Control, go to **Administration > Types**.
2. In the **Association Types** tab, select the association type's link that you want to modify.
3. Examine or modify the properties on the **Edit Association Type** dialog box as required.
4. Click **OK**.

Deleting an Association Type

When you attempt to delete an association type, keep the following points in mind:

- You cannot delete the association types that CentraSite provides out-of-the-box (not even if you belong to a role with the `Manage Asset Types` permission).
- You can delete an association type *only if there is no relationship attribute defined with it in the catalog*. After a association type has been assigned to a relationship attribute, it can no longer be deleted.
- You cannot delete the stand-alone association type that is in use for defining an association in the catalog.

Deleting a Single Association Type

To delete an association type

1. In the CentraSite Control, go to **Administration > Types**.
2. Go to the **Association Types** tab.
3. Enable the checkbox next to the name of an association type that you want to delete.
4. Click **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Deleting Multiple Association Types in a Single Step

You can delete multiple association types in a single step. The rules described above for deleting a single association type apply also when deleting multiple association types.

Important: If you have selected several association types, you can use the **Delete** button to delete the types. However, you are not allowed to delete types for which you do not have the required permission.

To delete multiple association types in a single operation

1. In CentraSite Control, go to **Administration > Types**.
2. Go to the **Association Types** tab.
3. Mark the checkboxes of the types that you want to delete.

4. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

7 Customizing Lifecycle Management

Overview of Customizing Lifecycle Management

CentraSite provides the ability to define and track the lifecycle of an asset by using a state model. CentraSite allows the use of active policies to govern specific transitions in the lifecycle management process.

Lifecycle models can theoretically be complex and go across multiple stages (machines). In CentraSite, the stages are called nodes. A lifecycle node is a machine within a lifecycle management model.

Lifecycle models can be applied to:

- Assets
- Policies
- Lifecycle models

Note: CentraSite provides predefined lifecycle models for several types of objects. For information about these lifecycle models, see "The Predefined Lifecycle Models Installed with CentraSite" on page 326.

Overview of Lifecycle Management

Lifecycle management is of importance to every enterprise that wants to implement a process-driven SOA with emphasis on adaptability, service reuse and improvement. The lifecycle management (LCM) system for CentraSite helps to:

- Assess change impact and manageability across all service consumers;
- Ensure service quality through an integrated lifecycle approval process;
- Enable a single viewpoint for service stages and their artifacts.

Typically, assets such as web services pass through different states (designing, implementation, testing) before they can be used in a production environment. As the number of objects in a registry grows, it is necessary to introduce stages (Development, Test, Production) to provide adequate operational environments for all parties involved in the lifecycle. Furthermore, an organization may want to add conditions and rules for passing an object through the lifecycle. Therefore the registry should allow administrators to define roles and permissions and connect these to lifecycle steps.

Lifecycle Models

A lifecycle model is a state machine that can be maintained in CentraSite and applied to an asset type. In CentraSite, the lifecycles of assets, policies and also lifecycle models themselves can be managed. Lifecycle models are defined by administrators for their organization.

There are two levels of lifecycle models: system-wide models and organization-specific models. A system-wide model applies to objects that can be owned by any organization, whereas an organization-specific model applies only to objects that are owned by a specific organization. System-wide models have precedence over organization-specific models; when a system-wide model for an object type (set of types) exists, it takes priority over all other models. However, if an organization-specific model already exists and a system-wide model is added, then the organization-specific model still exists and the objects that are in this model will complete their lifecycle without effects. Only new objects will be assigned to the new system-wide model.

Each organization-specific lifecycle model belongs to an organization. System-wide models do not belong to any organization, but apply to all organizations. System-wide models may have organization-specific policies attached to them.

For a given object type (say `Service`), each organization can define and activate its own lifecycle model, so that there are several models that can control the `Service` type. However, per organization only one lifecycle model can be active. When a new `Service` instance is created, then the creating user belongs to an organization, and that organization's active lifecycle model will be used to control the particular `Service` instance.

It is possible to define a lifecycle model which consists of multiple nodes (registries) to make up one overall model. Each node only knows the model for the current node. It also knows the nodes that make up the complete lifecycle model and where they are.

Global models may have organization-specific policies attached to them. Organizational models are technically hierarchical, as organizations may contain sub-organizations.

A lifecycle model can itself be assigned to a lifecycle model, i.e. to enable a lifecycle model it may be necessary to go through a lifecycle model to activate it for usage.

Lifecycle management is possible for the object types that are classified as assets, as well as for policies and lifecycle models themselves.

The following points should also be noted:

- Organization-specific models are defined by the administrator of the organization.
- An object type may have more than one lifecycle model defined, but only one per organization can be defined as active at any given time.

Stages for Lifecycle Registries

Most software engineering methodologies define fundamental phases, that start with requirements or needs for a software and end with a productive software. These phases apply to typical lifecycle stages like Analysis, Development, Test, Production. In general these stages are represented by dedicated software and hardware environments like server, repositories and tools. A registry adds benefit to all stages as it references, collects and maintains structured information of relevance in each of these stages. Especially for enterprises following the service oriented architecture, the CentraSite provides registry stages that can be used to take strong control over the lifecycle process the on one hand but allow for loosely coupled stage environments on the other hand.

If we take the example of the stages Architecture, Development, Test and Production, the following tasks could apply to them:

- **Architecture.** If services are developed in a top-down approach, the Architecture registry keeps track of all planning and preparation information. A service request triggers design and approval steps that may lead to the implementation or rejection of the requested service. During the service design process, the service interface as well as metadata may undergo changes. Only the final approval by an authorized role marks the service as ready for implementation. Examples of metadata which may be of special interest in the Architecture stage are:
  - Organization and contact responsible for service provisioning;
  - Date planned to go live;
  - Planned service consumer; dates planned to consume the service.
- **Development.** For the implementation of the service, it is important that only approved service descriptions are used. Service consumers often start to implement their service usage at the same time and rely on the service description. Multiple instances of Development registries may exist. These registries may be volatile, because development environments exist for the duration of one release and can be set up in a different configuration. Therefore it must always be possible to set up a development registry with the services that are of interest for the projects using this environment.
- **Test.** In loosely coupled service oriented architectures, the test laboratory often integrates for the first time different service participants which previously worked in dedicated environments. Furthermore the test cycle requires the repeated handover from development to test. The registries can support these dynamics by providing easy-to-use export and import facilities for all service related information and components. As the number of service provider and consumer applications grows, the test environments also get volatile and its registries need the flexibility to be adapted to the different combinations of test participants.
- **Production.** Finally, the registry in the production environment is the single point of truth for all applications requesting service lookup. Objects should enter this registry only if they have passed defined approval steps and if they are from a prescribed origin, for example the test registry. Metadata in the registry can inform about contacts in case of malfunction. Or the registry can be used to notify contacts in case of planned restrictions in availability.

Given these different tasks and requirements, it is obvious that multiple registries become important for optimal support of specific needs; while the Architecture is interested in services, service versions and participants still in planning, the Production is not. Test registries keep objects and references with physical properties - like endpoints - that are only valid during the test phase and are of no interest either to Architecture or Production. However, it is important that defined transitions exist from one stage to another and that each object can be traced to its current stage.

This four-stage topology is an example for enterprises with a strong top-down approach in their enterprise architecture; simpler or more complex topologies are also possible.

Note: This four-stage topology is an example for enterprises with a strong top-down approach in their enterprise architecture; simpler or more complex topologies are also possible. In practice, any deployment involving more than two stages leads to significant administrative overhead involved in managing multiple registries (in particular, creating LCMs and policies across all instances) and promoting objects from one registry to another.

Defining a Lifecycle Model in CentraSite

When a lifecycle model is stored in CentraSite, it must be a valid state machine. This specifically means that all of the following semantic rules must be checked by CentraSite:

- There is an initial state. All objects under the control of the lifecycle model are initially in this state.
- Each state must be reachable from at least one other state.
- A default next state can be defined for each state.
- There must be at least one end state. An end state is one that it does not have any next state.
- There may be a preferred transition from one state to the next (for the UIs to use).

States for Registry Objects

The registry stages are broken down into states for lifecycle-enabled objects like services. This way the lifecycle process can be split into smaller steps, each supported and controlled by its responsible stage registry. States are connected to each other by allowable transitions. An approval process can define the conditions and activities to set a service from one state to another. CentraSite comes with the following:

- A set of object states, associated to each registry stage
- A state-transition proposal to control the lifecycle
- An example for collaboration management by roles, rights and notifications

By using stages and states, you ensure that the registry object always has one defined state, even if multiple registries exist. State transitions are restricted; this ensures that the registry object had passed the lifecycle process correctly and achieved the required quality.

When moving from one state to the next, the following situations can exist:

- The required next state is in the same stage as the current state. In this case, the state change can be performed directly on the registry object from the user interface.

- The required next state is not in the same stage as the current state. In this case, the object must be exported (where necessary, along with its associated objects). In the exported archive, the current state remains unchanged. On importing the archive to the new stage, the state is set automatically to the appropriate new state.

Applying Lifecycle Models to Asset Types

You can associate a lifecycle model with one or more asset types. If an active lifecycle model is already associated with a particular type within an organization, you cannot assign another lifecycle model to that type. In other words, a particular object type can be associated with one and only one lifecycle model within a particular organization. If you want to switch an object type to a different lifecycle model, you must disassociate it with the current model and then assign it to the new model.

Applying Lifecycle Models to Virtual Types

You can configure a virtual type to follow the same lifecycle model as its base type or you can give the virtual type its own lifecycle model to follow. Whether a virtual type follows the lifecycle model of its base type is determined by the type's **Inherit Base Type LCM** setting.

When the **Inherit Base Type LCM** option is enabled for a virtual type, the virtual type automatically inherits its lifecycle model from the base type if the virtual type does not have an assigned lifecycle model of its own. (In other words, the lifecycle model for the base type serves as the default lifecycle model for the virtual type. CentraSite applies the lifecycle model of the base type to the virtual type only when the virtual type has no other lifecycle model assigned to it.)

If the **Inherit Base Type LCM** option is disabled, the lifecycle of the virtual type is completely independent of the lifecycle of the base type. The virtual type will only have lifecycle model if you explicitly assign one to it.

The following table summarizes how lifecycle model is applied to a virtual type depending on the state of the **Inherit Base Type LCM** option.

If the "Inherit Base Type LCM" option is...	And the Base Type...	And the Virtual Type...	Instances of the Virtual Type...
ENABLED	HAS an assigned lifecycle model	DOES NOT HAVE an assigned lifecycle model	Will follow the lifecycle model assigned to the Base Type
ENABLED	HAS an assigned lifecycle model	HAS an assigned lifecycle model	Will follow the lifecycle model

If the "Inherit Base Type LCM" option is...	And the Base Type...	And the Virtual Type...	Instances of the Virtual Type...
			assigned to the Virtual Type
ENABLED	DOES NOT HAVE an assigned lifecycle model	DOES NOT HAVE an assigned lifecycle model	Will not have an assigned lifecycle model
ENABLED	DOES NOT HAVE an assigned lifecycle model	HAS an assigned lifecycle model	Will follow the lifecycle model assigned to the Virtual Type
DISABLED	HAS an assigned lifecycle model	DOES NOT HAVE an assigned lifecycle model	Will not have an assigned lifecycle model
DISABLED	HAS an assigned lifecycle model	HAS an assigned lifecycle model	Will follow the lifecycle model assigned to the Virtual Type
DISABLED	DOES NOT HAVE an assigned lifecycle model	DOES NOT HAVE an assigned lifecycle model	Will not have an assigned lifecycle model
DISABLED	DOES NOT HAVE an assigned lifecycle model	HAS an assigned lifecycle model	Will follow the lifecycle model assigned to the Virtual Type

Note: If a virtual type initially inherits its lifecycle model from the base type and it later it is assigned its own lifecycle model, instances of the virtual type that were created while the type was following the lifecycle model of the base type will continue following that model. Only instances of the virtual type that are created after the new lifecycle model is assigned to the virtual type will comply with the virtual type's newly assigned lifecycle model.

For more information about virtual types and the **Inherit Base Type LCM** option, see the "What Is a Virtual Type?" on page 222. For information about which predefined types in CentraSite are virtual types, see "Predefined Asset Types in CentraSite" on page 263.

Who Can Create and Manage Lifecycle Models?

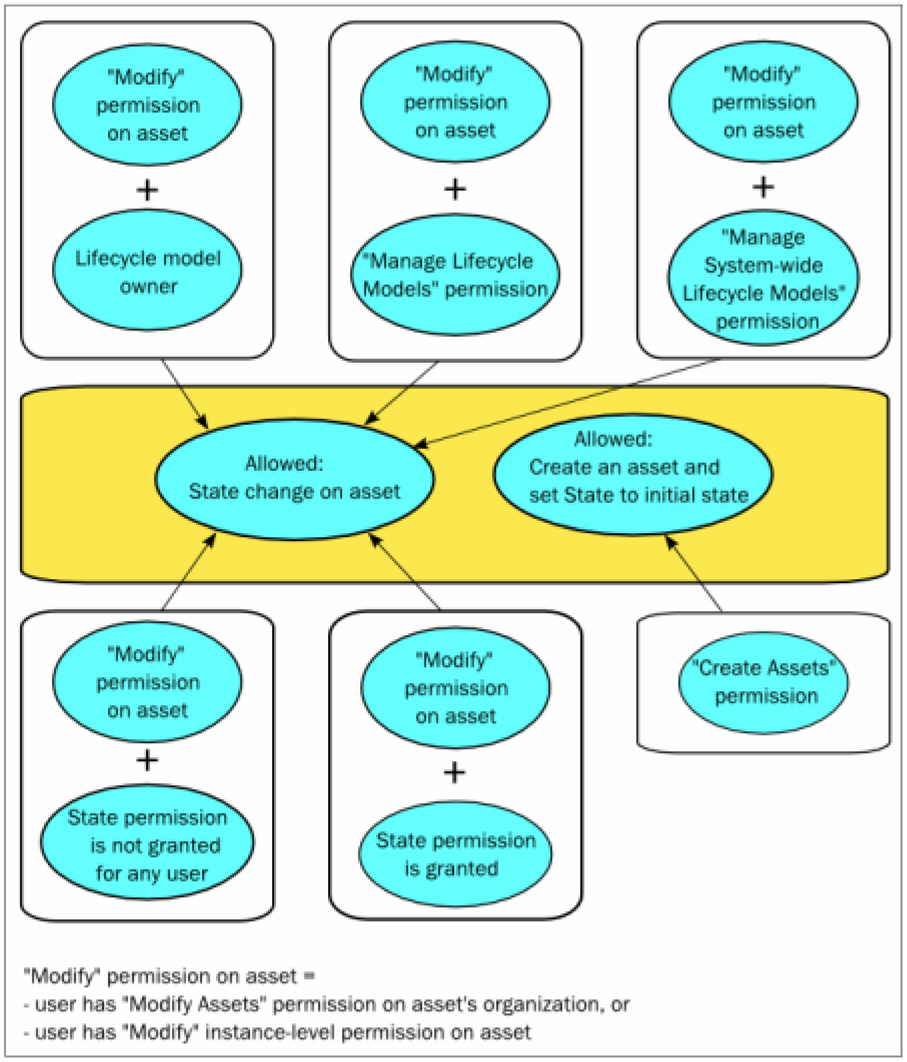
To create and manage (i.e., view, edit, and delete) lifecycle models for an organization, you must belong to a role that has the Manage Lifecycle Models permission for the organization.

To create and manage *system-wide* lifecycle models (i.e., lifecycle models that apply to all organizations), you must belong to a role that has the Manage System-wide Lifecycle Models permission.

To be able to activate an organization-specific lifecycle model or a system-wide lifecycle model, you must have the Modify Assets permission for the organization(s) that own the lifecycle model's assigned asset types. This is because when a lifecycle model is activated, assets of its assigned asset types in the organization(s) will be set to the initial state of the lifecycle model, and this requires the modify permission for the assets.

To be able to delete an organization-specific lifecycle model or a system-wide lifecycle model, you must have the Modify Assets permission for the organization(s) that own the lifecycle model's assigned asset types. This is because when a lifecycle model is deleted, assets of its assigned asset types in the organization(s) will be updated to remove the reference to the lifecycle model, and this requires the modify permission for the assets.

Several rules determine who can change the lifecycle state of an asset. These rules are summarized in the following diagram.


"Modify" permission on asset
+
Lifecycle model owner
"Modify" permission on asset
+
"Manage Lifecycle Models" permission
"Modify" permission on asset
+
"Manage System-wide Lifecycle Models" permission
Allowed: State change on asset
Allowed: Create an asset and set State to initial state
"Modify" permission on asset
+
State permission is not granted for any user
"Modify" permission on asset
+
State permission is granted
"Create Assets" permission
"Modify" permission on asset =
- user has "Modify Assets" permission on asset's organization, or
- user has "Modify" instance-level permission on asset

For example: If you are the owner of a lifecycle model, you can assign any lifecycle state of this lifecycle to an asset whose asset type has this lifecycle model assigned, as long as you have the Modify Assets permission for the asset.

Any user who has the Create Assets permission can create an asset whose asset type has a lifecycle model assigned. When the asset is created, CentraSite automatically sets the lifecycle state of this asset to the initial state. However, to change the state from the initial state to another lifecycle state, the user requires the appropriate permissions as described above.

Note that the Manage Assets permission does NOT include the rights to change lifecycle states.

For more information about roles and permissions, see "About Roles and Permissions" on page 115.

Viewing or Editing a Lifecycle Model

Use the following procedures to view or edit the properties of a lifecycle model.

Viewing and/or Editing a Lifecycle Model

To view and/or edit a lifecycle model

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.

   By default, the names of all of the available lifecycle models are displayed.

   If you want to filter the list to see just a subset of the available lifecycle models, type a partial string in the **Search** field. CentraSite applies the filter to the **Name** column. The **Search** field is a type-ahead field, so as soon as you enter any characters, the display will be updated to show only those lifecycle models whose name contains the specified characters. The wildcard character `%` is supported.
2. In CentraSite Control, go to **Administration > Lifecycles > Models**.
3. Locate the lifecycle model that you want to view or edit.
4. From the lifecycle model's context menu, select **Details**.
5. View or edit the attributes on the Edit Lifecycle Model page as required.
6. Click **Save**.

Viewing Multiple Asset Types

To view multiple asset types

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. Mark the checkboxes of the lifecycle models whose details you want to view.
3. In the **Actions** menu, click **Details**.

   The **Details** view of each of the selected lifecycle models is now displayed.

Creating a Lifecycle Model

The definition of a lifecycle model consists of specifying basic attributes of the model such as its name and owning organization, the model's states and transitions, user permissions, and the associated asset types and policies.

Defining the Basic Attributes of the Model

To define the basic attributes of the model

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. Click **Add Model**.
3. In the field **Name**, enter the name of the lifecycle model.
4. In the field **Description**, you can optionally add a comment that describes the purpose of the lifecycle model.
5. In the field **Version**, you can supply a user-defined version number for the model.
6. In the **Organization** drop-down box, choose the organization for which the lifecycle model applies.

   The organization names displayed in the drop-down box are the organizations for which you have the Manage Lifecycle Models permission. If you select one of these names, the lifecycle model applies just to objects belonging to the selected organization.

   If you have the Manage System-wide Lifecycle Models permission, you will be able to select the `All` option, meaning that the lifecycle model is a system-wide lifecycle model.

   Important: Set the system-wide property (`All`) with care. You cannot change this assignment after the lifecycle model is created.

Defining the States and State Transitions of the Model

To define the states and state transitions of the model

1. Click the **States** tab, if it is not already selected.
2. Click **Add State**.
3. In the field **State Name**, supply the name of your state.
4. In the field **Description**, you can optionally enter a comment that describes the purpose of the state.
5. In the area labeled **Transitions**, specify a state than can be reached as a result of a transition from the current state. In the column **Target State**, you can select the target state from a drop-down list containing all of the states defined so far.

   Note: You may find it more convenient to define all of the states before you start to define the state transitions.

If you wish to define more than one target state from the current state, select the + icon to create a new line.

If you have more than one target state, select one of the target states as the default target state by choosing the radio button in the column labeled **Default**.

You can show or hide the transition area for a state by choosing the triangle icon in the header line of the **Transitions** area.

6. Use the **Add State** button to define additional states as required.

Associating an Asset Type with the Model

When a lifecycle model is assigned to an asset type, and the lifecycle mode is activated (i.e. set to the state `Productive`), all existing assets of this type in the user's organization are assigned to the first state of the lifecycle model. Also, each subsequently created asset of the given asset type in the user's organization will be automatically associated with the lifecycle model and will be initially set to the first state of the lifecycle model.

An asset type can only be associated with one lifecycle model at a time. If a lifecycle model includes an asset type that is already associated with another lifecycle model, an error message will be displayed when you try to activate the lifecycle model.

If you need to add another asset type to a lifecycle model that is already in the `Productive` state, you must create a new version of the lifecycle model and add the asset type to the new version. For information about creating a new version of a lifecycle model, see "Updating a Lifecycle Model" on page 322.

To assign a lifecycle model to an asset

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. In the list of lifecycle models displayed, click the required lifecycle model to display its details page.
3. Select the **Associated Types** tab. This displays a list of available asset types that can be associated with the lifecycle model.
4. To define that a particular asset type should be associated with the lifecycle model, select the appropriate asset type from the list shown in the box **Available Types**, then use the arrow buttons to copy the asset type into the box **Selected Types**.

Asset types that are already associated with another lifecycle model are displayed with the name of the lifecycle model beside them. For example, if the asset type Application is already associated with the lifecycle model MyModel, this will appear as `Application (MyModel)`.

Asset types that are associated with a system-wide lifecycle model will only appear in the list of available asset types if you have the Manage System-wide Lifecycle Models permission.

Note: If you apply a lifecycle model to an asset type that has virtual types associated with it, the model will automatically be applied to those virtual types that meet both of the following conditions:

- The virtual type's **Inherit Base Type LCM** option is enabled.
- The virtual types does not have a lifecycle model assigned to it.

For more information about how CentraSite applies lifecycle models to base types and virtual types, see "Applying Lifecycle Models to Virtual Types" on page 312.

Defining the User and Group Permissions Associated with the States

Now you define the user and group permissions associated with each state of the lifecycle model. Each permission defines whether a user or group can move assets into a particular lifecycle state. Such permissions are referred to as *state permissions*.

If you leave the list of users and groups empty, CentraSite grants state permissions to all users and groups. If the list contains at least one user or group, permissions are denied for all other users and groups who are not in the list.

To define the user and group permissions associated with the model

1. Start by selecting the **State Permissions** tab.
2. Click **Add Users/Groups** to display the list of currently defined users and groups.
3. From the list of displayed names, choose a user or group for whom you wish to grant or deny permission to change the lifecycle state of an asset.
4. Select the states into which the user or group is allowed to move an asset. You can either check the checkboxes for the states individually, or select them all by checking the corresponding box in the **ALL** column. To deselect a state that is already selected, uncheck its checkbox.

   Note that there is no column corresponding to the initial state of an asset, because all assets under control of this lifecycle model are automatically put into the initial state.
5. Repeat the previous steps for additional users or groups as required.

Defining Policies Associated with the Model

To define policies associated with the model

1. Finally, you can define that one or more policies will be triggered when a state in the lifecycle is entered.

   Before proceeding, click **Save** to ensure that the model that you have defined so far validates (i.e. contains no inconsistencies).
2. Return to the **States** tab and click the **Policies** button for the required state. This displays a list of the currently defined policies for the state, and whether or not the policies are currently active or inactive.

A policy can be assigned to a particular state when you create a policy. For more information working with design/change-time policies, see the *CentraSite User's Guide*.

If you want to see the details of any of the given policies, select the policy name. This opens the **Policy Details** dialog.

Creating Lifecycle Stages

A lifecycle model can contain one or more stages. Typically, stages are used to represent a clearly-defined deployment scenario within an object's development cycle. You could, for example, decide to use the lifecycle model during various stages of a product lifecycle such as `development`, `test` and `production`. The individual stages are also usually deployed on different physical machines, which is normal practice if for example you have a test environment and a production environment. Each stage you define contains all of the states of the model.

To create a lifecycle stage

1. In CentraSite Control, choose **Administration > Lifecycles > Stages**.
2. Click **Add Stage**. The **Add Stage** dialog opens.
3. Enter values for the dialog fields as follows:

Fields Name	Contents
Name	The stage name. Here you can enter any text string as the name of the stage.
Description	(Optional) A description of the purpose of the stage.
Host Name	The name of the HTTP host machine where the lifecycle stage is deployed on a CentraSite installation.
Host Port	The HTTP port number of the CentraSite installation on the host machine where the lifecycle stage is deployed.
Use SSL	Mark this checkbox if the communication with the specified host machine should use the SSL protocol for secure communication.

4. Click **OK**.

Defining Stage Transitions

After you have defined your states and state transitions, you can define a stage transition from an end state of the current stage to one or more new stages. An end state is a state that has no transitions to other states.

The term *promotion* is used to describe the transition of an asset from one stage to the next stage.

To define stage transitions

1. In the details page of the lifecycle model, click the **Stages** button beside the appropriate state name.
2. From the list of stage names displayed, select the stage or stages that can be the target of a transition from the selected state.
3. Click **OK**.

Validating and Saving the Model

When you have completed all of the above steps, click **Save**. When you do this, the model is validated. If there are no validation errors, the lifecycle model is stored. If any part of the model is invalid, this will be indicated by an appropriate error message.

A model is valid if:

- it consists of at least one state, and
- there is a possible transition path to all of the states except the first state.

Activating a Lifecycle Model

Lifecycle models can be applied not only to assets but also to other lifecycle models. CentraSite provides a default lifecycle model that is applied automatically to all user-defined lifecycle models. This default lifecycle model defines the following states:

State	Meaning
New	A user-defined lifecycle model has been saved but is not yet been activated for use with its associated asset types.
Productive	A user-defined lifecycle model has been activated for use with its associated asset types. The lifecycle status of assets of the associated asset types is visible in the detail view of the asset instances.
Retired	The lifecycle model is no longer in use and cannot be reactivated.

When you create a user-defined lifecycle model and assign it to one or more asset types, the lifecycle model is initially inactive (i.e. in the state `New`).

Important: To be able to activate a system-wide lifecycle model, you must have the Modify Assets permission for all organizations that own the lifecycle model's assigned asset types. This is because when a system-wide lifecycle model is activated, assets of its assigned asset types from all organizations will be set to the initial state of the lifecycle model, and this requires the modify permission for the assets.

To activate a lifecycle model

1. Open the Edit Lifecycle Model page for the lifecycle model, if it is not open already.

   You reach this page by choosing **Administration > Lifecycles > Models**, then choosing the appropriate model from the displayed list.
2. Click **Change State**.
3. Select the state `Productive`, then click **OK**.

While a lifecycle model is in the `Productive` state, you cannot modify it. If you need to modify a lifecycle model, you must create a new version of it.

Updating a Lifecycle Model

To update a lifecycle model, you create a new version of the model, and in the new version you can add states, remove states, rename states, change the relationships between the states and change the list of assigned asset types.

A lifecycle model can only be updated when it is in the `New` state.

When a lifecycle model is in the `Productive` or `Retired` state, you cannot change the state of the current version of the model back to `New`; in this case, you can only reach the `New` state by creating a new version of the lifecycle model.

To update a lifecycle model

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. Locate the name of the lifecycle model that you want to update.
3. Create a new version of the lifecycle model by choosing **Create new version** in the context menu.

   When you create the new version, it is not yet activated; the lifecycle state of the new version is `New`, and the lifecycle version of the previously used version is still `Productive`.
4. Make any changes you require to the new version of the lifecycle model.
5. Set the new version of the lifecycle model to the `Productive` state.

This automatically changes the state of the previously used version from `Productive` to `Retired`.

All existing assets of the asset types that use this lifecycle model are automatically set for use with the new version.

When you change a lifecycle model that is already in use, you must ensure that all of the states that were in use in the old model are also available in the new model. If the old state model contains states that no asset instance is currently using, these states do not need to be present in the new model. The state transitions in the new model do not depend on the state transitions in the old model; you can define the state transitions in the new model as you please.

Deleting a State from a Model

A state can only be deleted if there is currently no policy that is triggered when this state is entered.

To delete a state from a model

1. Display a list of available lifecycle models. If you need procedures for this step, see "Viewing or Editing a Lifecycle Model" on page 316.
2. Locate the required lifecycle model from the list and open its detail view.
3. In the detail view, select the **States** tab if it is not already selected.
4. Mark the checkbox of the state you wish to delete.
5. Click **Delete**.
6. In the other states of the model, remove all transitions to the deleted state.
7. Save your changes.

Deleting a Lifecycle Model

If you wish to delete a lifecycle model, you must first deactivate the lifecycle model. This means that you should change the lifecycle model's own lifecycle status from `Productive` to `Retired`.

If several versions of the lifecycle model exist, CentraSite offers two commands for deleting versions, namely Delete and Purge. The Delete command deletes the newest version of the lifecycle model; if you use the Delete command on an older version, the command will be rejected. The Purge command deletes the requested version and all other versions that are older than the requested version.

Command	Description
Delete	Deletes the newest version of a lifecycle model. This command cannot be used on an older version of the lifecycle model.
Purge	Deletes the selected version and all older versions of a lifecycle model. If the selected version is the newest version, this command removes the lifecycle model altogether from CentraSite.

When you purge old versions of a lifecycle model, any existing newer versions will not be deleted. In this case, assets governed by this lifecycle model will continue to be governed by this lifecycle model.

If you have several versions of a lifecycle model and one of the older versions is still in the `Productive` state, you cannot purge any newer version, since this would automatically try to delete the older version that is in `Productive` state. This can happen, for example, if you have just created a new version of a lifecycle model but have not yet set the new version to the `Productive` state.

If you purge the newest version of a lifecycle model, you automatically delete all of the older versions also, i.e. you remove the lifecycle model altogether from CentraSite. In this case, all assets that were governed by the lifecycle model are now of course no longer governed by the lifecycle model. You do not need to adapt these assets in any way for non-lifecycle usage, and they are treated by CentraSite as if they had never been governed by a lifecycle model. The only visible change is that when you display the detail view of such an asset, the lifecycle status of the asset is no longer displayed.

Note: The lifecycle model for lifecycles and the lifecycle model for policies cannot be deleted. Also, if several versions of such a lifecycle model exist, none of the versions can be purged.

Deleting the Most Recent Version of a Lifecycle Model

Use the following procedure to delete the most recent version of a lifecycle model.

To delete the most recent version of a lifecycle model

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. If the lifecycle model that you wish to delete is currently in `Productive` mode, select the entry **Change Lifecycle State** from the context menu and change the state to `Retired`.
3. Mark the checkbox for the required version of lifecycle model.
4. In the context menu, click **Delete**.

   The selected version will be deleted; all older versions will not be affected.

Deleting a Version and All Older Versions of a Lifecycle Model (Purging)

Use the following procedure to delete a version and all older versions of a lifecycle model.

To delete a version and all older versions of a lifecycle model (i.e. to purge)

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. If any version of the lifecycle model that you wish to purge is currently in `Productive` mode, select the entry **Change Lifecycle State** from the context menu of the version and change the state to `Retired`.
3. Mark the checkbox for the version of lifecycle model where you want to begin purging.
4. In the context menu, click **Purge**.

   The selected version and all older versions will be deleted; any newer versions will not be affected.

Deleting Multiple Lifecycle Models in a Single Operation

You can delete multiple lifecycle models in a single step. The rules described above for deleting a single lifecycle model apply also when deleting multiple lifecycle models.

Important: If you have selected several lifecycle models where one or more of them are predefined models, you can use **Delete** to delete them. However, as you are not allowed to delete predefined lifecycle models, only models you have permission for will be deleted. The same applies to any other lifecycle models for which you do not have the required permission.

To delete multiple lifecycle models in a single operation

1. In CentraSite Control, go to **Administration > Lifecycles > Models**.
2. Mark the checkboxes of the lifecycle models that you want to delete.
3. In the **Actions** menu, click **Delete**.

Lifecycle Model for Lifecycle Models (LCM for LCMs)

CentraSite provides a lifecycle model for lifecycle models, which consists of three states: `New`, `Productive` and `Retired`. Predefined policies associated with the LCM for LCMs activate and deactivate a lifecycle model as appropriate when you switch the lifecycle model's lifecycle state. The model is also structured in a way that allows CentraSite to automatically deactivate an old version of a lifecycle model when a new one is activated.

Normally, there is no need to change the LCM for LCMs, and you should only consider modifying it if you have a compelling reason to do so. Because of the complex nature of the LCM for LCMs and its associated policies, any changes you make should be limited to the ones described below.

You cannot change the original version of the predefined LCM for LCMs, but you can create a new version of the LCM for LCMs and modify the new version while it is still in the `New` lifecycle state. Then set the lifecycle state of the new version to `Productive`, which activates the new version and deactivates the old version.

Here are the changes you can make in a new version of the LCM for LCMs before you set its lifecycle state to `Productive`:

- You can edit the lifecycle model's name, description, and permission settings.
- You can rename the states in the lifecycle model.
- You can associate additional policies with the states in the lifecycle model. (Do not modify or delete the predefined policies that are associated with this lifecycle model, however.)

Do not add states to the model, and do not remove states from the model. Also, do not modify any of its state transitions.

The Predefined Lifecycle Models Installed with CentraSite

The following table lists the predefined lifecycle models that are installed with CentraSite.

CentraSite uses some of these lifecycles to govern particular types of registry objects (Policy objects, for example). These lifecycle models have system policies associated with them. You can customize these lifecycle models in only limited ways. These lifecycle models are already active when CentraSite is installed.

Other lifecycle models are treated as "user-defined" lifecycle models. These lifecycle models are provided to you as a convenience. You can use them instead of creating your own models from scratch. You may customize these lifecycle models in any way you like. These lifecycle models are not active when you install CentraSite. If you want to use them, you must activate them.

Model Name	Applies to	Active upon Installation of CentraSite?	Customizable?
Lifecycle Model for Lifecycles	Lifecycle Models objects	Yes	In limited ways. For more information, see "Lifecycle Model for Lifecycle Models

Model Name	Applies to	Active upon Installation of CentraSite?	Customizable?
			(LCM for LCMs)" on page 325.
Policy Lifecycle	Policy objects (Both design/ change-time policies and run-time policies)	Yes	In limited ways. For more information, see the *CentraSite User's Guide*.
BPM Process Lifecycle	Process objects	No	Yes
Service Lifecycle	Service assets	No	Yes

8 Taxonomies

Introduction

A *taxonomy* categorizes assets in CentraSite so that a consumer can search for assets within a particular category. CentraSite provides several predefined taxonomies, which are available to all users. Additionally, you can create custom taxonomies to suit your business needs and subdivide them by creating *categories*. Categories help consumers locate assets more easily. For example, if you are offering assets to help your consumers better manage their finances, classifying your assets under `personal banking` or `money management` might help them locate your assets more easily.

Who Can Create and Manage Taxonomies?

To create taxonomies, you must belong to a role that has the Manage Taxonomies permission. Besides allowing you to create taxonomies, this permission allows you to edit and delete any user-defined taxonomy (the predefined taxonomies provided by CentraSite can be modified in certain ways, but cannot be deleted). By default, users in the CentraSite Administrator or Asset Type Administrator role have this permission, although an administrator can grant this permission to other roles.

If you do not belong to a role that includes the Manage Taxonomies permission, you can edit and delete a taxonomy if:

- You have the appropriate instance-level permissions (i.e., Modify or Full permission) on the taxonomy.

  —AND—

- You belong to a role that includes the Use the Administration UI permission.

For more information about roles and permissions, see "About Roles and Permissions" on page 115.

Visibility of Taxonomies in CentraSite

All users have implicit (and irrevocable) view permission on taxonomies. Broadly speaking, this permission enables any user to access any taxonomy for the purpose of classifying or filtering registry objects. However, taxonomies have the following additional properties that control whether they are visible to users within the CentraSite Control user interface.

Property	Description
Taxonomy is Browsable	Determines whether the taxonomy appears in the taxonomy lists that CentraSite Control displays to users for the purposes of classifying or filtering objects. If this property

Property	Description
	is enabled for a taxonomy, CentraSite Control includes the taxonomy in the following lists: ■ The **Browse by** list in the **Asset Catalog > Browse** page. ■ The **Add Classification** dialog box displayed from the **Classification** profile. The **Taxonomy is Browsable** property is an attribute that you set when you create a taxonomy. You might disable this property, for example, if a taxonomy is intended to be used to classify objects programmatically (e.g., using a policy). Doing this will prevent end users from using the taxonomy to classify assets using the **Classification** profile in CentraSite Control. You can also use this property to suppress the display of the predefined taxonomies that CentraSite installs. For example, if your site has no need to use the NAICS taxonomies provided by CentraSite, you can eliminate them from the taxonomy lists displayed in CentraSite Control by disabling their **Taxonomy is Browsable** property.
Applicable to Object Types	When a taxonomy is browsable (i.e., when its **Taxonomy is Browsable** property is enabled), the **Applicable to Object Types** property determines for which object types the taxonomy will be shown. For example, if you specify that a taxonomy is not applicable to XML Schemas, CentraSite Control will not include the taxonomy in the taxonomy lists that users see when they add classifiers to XML Schemas.
Internal	Indicates whether a taxonomy is meant to for use by end users. Taxonomies that are *internal* are designed to support CentraSite's own internal processes and are, therefore, suppressed from most taxonomy lists displayed in CentraSite Control. A taxonomy's Internal property cannot be assigned or viewed through CentraSite Control. It can only be accessed through the API.

Creating a Taxonomy

Perform these steps to create a taxonomy and save it to CentraSite.

To create a taxonomy

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, click **Add Taxonomy**.
3. In the **Add Taxonomy** page, specify the following fields:

In this field...	Specify...
Name	A name for the taxonomy. Be aware that this is the name that users will see when they search for assets using the Browse catalog page. The name should be meaningful (for example, `Software AG ASIA-PAC`, not `SAG-AP`). **Note:** A taxonomy name does not need to be unique within CentraSite. However, to reduce ambiguity, you should avoid giving multiple taxonomies the same name.
Description	*Optional*. A descriptive comment for the taxonomy.
Documentation	*Optional*. Specify a file that contains additional external documentation for this taxonomy. You can use the **Browse** button for this field to locate the file.
Taxonomy is browsable	*Optional*. Whether you want the taxonomy to be visible to users for filtering and classification purposes in CentraSite Control. For more information about this property, see "Visibility of Taxonomies in CentraSite" on page 330.
Icon	*Optional*. The location of a bitmap file that contains an icon that will be displayed in the user interface to identify this new taxonomy. The bitmap format must be either JPG, GIF or PNG; it must be 16x16 pixels, to match the size of the existing taxonomies.

4. Specify the object types that are allowed to use the taxonomy for association purposes. You do this by selecting the tab **Applicable to Object Types** and marking the check boxes of the required object types. For more information about associating object types to a taxonomy, see "Associating Taxonomy to an Object Type" on page 333.
5. To enable other users to view, modify and/or delete a taxonomy that you have created, you must modify the taxonomy's permission settings. You do this by selecting the tab **Permissions** and making the appropriate entries there. For more information about allowing other users to access a taxonomy, see "Setting Permissions on a Taxonomy" on page 334.

6. Click **Save** to add the new taxonomy to CentraSite.

Associating Taxonomy to an Object Type

You can associate a taxonomy to multiple object types. When you do this, you can classify objects of the specified type by the associated taxonomy.

To associate a taxonomy to an object type

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, click the name of the taxonomy that you want to associate with the object types.
3. In the Edit Taxonomy page, select the **Applicable to Object Types** tab. Select the object types that you want to associate with the taxonomy.

   When associating the taxonomy with object types, keep the following points in mind:

   - You can either mark the check boxes for the object types individually, or select them all by checking the **All Object Types** check box.
   - By default, CentraSite displays the **All Object Types** check box as `selected`. When this check box is selected, the check boxes for the list of object types are automatically displayed as `selected` and `disabled`.

     This default selection does not associate the taxonomy with the selected object types. Instead, it only allows the taxonomy to be applied to the selected object types (that is, CentraSite allows you to classify the selected object types using the taxonomy).
   - If you unmark the **All Object Types** check box, and mark ALL of the object types as `selected`, then there does not exist an association, and the taxonomy is applicable to all of the selected object types.
   - If you unmark the **All Object Types** check box and the list of ALL the object types, then there does not exist an association, and the taxonomy is still applicable to all the object types.
   - Therefore, to associate a taxonomy with one or more object types, you MUST:

     i. Unmark the**All Object Types** check box.

     ii. Manually select the check box(es) of the object types that you want to associate with the taxonomy.
4. Click **Save**.

Setting Permissions on a Taxonomy

All CentraSite users are permitted to view the taxonomies that you create. However, only you (as the owner of the taxonomy) and users who belong to a role with the Manage Taxonomies permission are allowed to modify or delete these taxonomies. To enable other users to modify and/or delete a taxonomy that you have created, you must modify the taxonomy's instance-level permission settings.

The following procedure describes how to set permissions on a taxonomy. When setting permissions on taxonomies, keep the following points in mind:

- To set permissions on a taxonomy, you must belong to a role that has the Manage Taxonomies permission or have the Full instance-level permission on the taxonomy itself.
- You can assign permissions to any individual user or group defined in CentraSite.
- The groups to which you can assign permissions include the following system-defined groups:

Group Name	Description
Users	All users within a specified organization.
Members	All users within a specified organization and its child organizations.
Everyone	All users of all organizations and child organizations, *including guest users* (if your CentraSite permits access by guests).

- If a user has multiple permission assignments, the user receives the union of all the assignments. For example, if group A has Modify permission on a taxonomy and group B has Full permission on the same taxonomy, users that belong to both groups will, in effect, receive Full permission on the taxonomy.

To assign permissions to a taxonomy

1. Display the details page for the taxonomy whose permissions you want to edit. If you need procedures for this step, see "Viewing or Editing the Properties of a Taxonomy" on page 336.
2. In the Edit Taxonomy page, select the **Permissions** tab.
3. To add users or groups to the **Users/Groups** list, do the following:
   a. Click **Add Users/Groups**.
   b. Select the users and groups to which you want to assign permissions.

If you want to filter the list, type a partial string in the **Search** field. CentraSite applies the filter to the **User/Group** column.

Examples

String	Description
`b`	Displays names that contain `b`.
`bar`	Displays names that contain `bar`.
`%`	Displays all users and groups.

c. Click **OK**.

4. Use the **View**, **Modify**, and **Full** check boxes to assign specific permissions to each user and group in the **Users/Groups** list as follows:

Permission	Allows the selected user or group to...
View	View the taxonomy. **Note:** Disabling this permission will not prevent a user from accessing the taxonomy. CentraSite implicitly grants users view permission on all taxonomies. The implicit permission granted by CentraSite is not revoked by disabling the **View** permission on this tab.
Modify	View and edit the taxonomy.
Full	View, edit, and delete the taxonomy. This permission also allows the selected user or group to assign instance-level permissions to the taxonomy.

5. Click **Save** to save the new permission settings.

Note: If you have given users Modify or Full permission on the taxonomy, and you want them to be able to work with the taxonomy in CentraSite Control, be sure the users belong to a role that has the Use the Administration UI permission.

Viewing the Taxonomy List

You use the Taxonomies page to view the list of taxonomies on CentraSite. By default, this list displays all "browsable" taxonomies that are defined. To display all taxonomies, enable the **Show all Taxonomies** option.

To view the taxonomies list

1. In CentraSite Control, go to **Administration > Taxonomies** to view the list of taxonomies.
2. Select the **Show all Taxonomies** check box to view the complete list of user accessible taxonomies. (This list does not include taxonomies that CentraSite uses to support its own internal processes and are not meant to be used or modified by end users.)
3. The Taxonomies page list provides the following information about each taxonomy:

Column	Description
Taxonomies	The name of the taxonomy.
Description	A short description about the taxonomy.

Viewing or Editing the Properties of a Taxonomy

You use the **Edit Taxonomy** page to examine and/or modify the properties of a taxonomy.

To modify a taxonomy

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, click the name of the taxonomy whose details you want to modify.
3. Examine or modify the properties on the Edit Taxonomy page as required.
4. Click **OK**.

Deleting a Taxonomy

Deleting a taxonomy permanently removes the taxonomy from the CentraSite. When you delete a taxonomy keep the following points in mind:

- Before you attempt to delete a taxonomy, you must delete all of the categories underneath it.
- You cannot delete a taxonomy if it is currently used to classify one or more objects.
- You are not allowed to delete the predefined taxonomies provided by CentraSite (not even if you belong to a role with Manage Taxonomies permission).

Deleting a Single Taxonomy

To delete a taxonomy

1. In the CentraSite Control, go to **Administration > Taxonomies** to display the taxonomies list.
2. Enable the checkbox next to the name of the taxonomy that you want to delete.
3. Click **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Deleting Multiple Taxonomies in a Single Operation

You can delete multiple taxonomies in a single step. The rules described above for deleting a single taxonomy apply also when deleting multiple taxonomies.

Important: If you have selected several taxonomies where one or more of them are predefined taxonomies, you can use the **Delete** button to delete the taxonomies. However, as you are not allowed to delete predefined taxonomies, only taxonomies you have permission for will be deleted. The same applies to any other taxonomies for which you do not have the required permission.

To delete multiple taxonomies in a single operation

1. In CentraSite Control, go to **Administration > Taxonomies** to display the taxonomies list.
2. Mark the checkboxes of the taxonomies that you want to delete.
3. From the **Actions** menu, choose **Delete**.

   When you are prompted to confirm the delete operation, click **OK**.

Adding a Category

You can subdivide a taxonomy by creating categories and subcategories. When creating categories, keep the following points in mind:

- The **Add Category** context menu is visible for the following conditions:
  - If the taxonomy has already been categorized with one or more child taxonomies (categories).
  - If the taxonomy does not have a child category and has NEVER been classified for any of its applicable object types.
- The **Add Category** context menu is NOT visible for the following conditions:
  - If the taxonomy does not include a child category and is classified for at least one of its applicable object types.

- If the taxonomy and it's category are classified for at least one applicable object type.
- If the taxonomy is related to any one of the lifecycle model.

To add a category

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, locate the custom taxonomy you want to categorize.
3. In the context menu of the taxonomy, choose **Add Category**.
4. In the **Add Category** dialog box, specify the following fields:

In this field...	Specify...
Name	A name for the category. Be aware that this is the name that users will see in the taxonomy hierarchy when they view assets using the Browse page. The name should be meaningful (for example, `Business Application Systems`, not `BAS`). **Note:** A category name does not need to be unique within the taxonomy. However, to reduce ambiguity, you should avoid giving multiple categories the same name.
Description	*Optional.* A descriptive comment for the category.
Icon	*Optional.* The location of a bitmap file that contains an icon that will be displayed in the user interface to identify this category. The bitmap format must be either JPG, GIF or PNG; it should be 16x16 pixels, to match the size of the existing categories.

5. Click **OK**.

   You can expand and collapse the display of the category names by choosing the triangle icon beside the taxonomy name.

Viewing or Editing the Properties of a Category

Use the following procedure to view or modify a category.

To modify a category

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, locate the taxonomy that contains the category you want to modify.

3. Ensure that the taxonomy's categories are visible by choosing the triangle icon beside the taxonomy name.
4. Locate the category that you wish to modify, and choose **Details** in its context menu.
5. In the **Edit Category** dialog box, change the category's details as required
6. Click **OK**.

Deleting a Category

Use the following procedure to delete a category from a taxonomy. When deleting a taxonomy's categories, keep the following points in mind:

- To delete a category, you must first delete all of the sub-categories underneath it.
- You cannot delete a category if objects are currently classified by the category.
- You cannot delete categories from the predefined taxonomies provided by CentraSite (not even if you belong to a role with Manage Taxonomies permission).

To delete a category

1. In CentraSite Control, go to **Administration > Taxonomies**.
2. In the **Taxonomies** page, locate the taxonomy that contains the category you want to modify.
3. Ensure that the taxonomy's categories are visible by choosing the triangle icon beside the taxonomy name.
4. Locate the category that you wish to delete, then choose **Delete**. You can select multiple categories for deletion.

   When you are prompted to confirm the deletion, click **OK**.

9 Logging

Overview of Logging

To effectively manage the CentraSite registry/repository, it is necessary to get feedback about the activity and performance of the registry/repository. The CentraSite registry/ repository provides comprehensive and flexible logging functionality for tracking design/change-time and run-time events in CentraSite Control. CentraSite stores the log and monitoring data of all registry objects (e.g., policies, assets, etc.) as log records in the CentraSite Log Database.

As your logs grow, you may want to purge log records to avoid performance degradation. CentraSite provides the ability to purge log records and back them up to another location. You can purge log records manually (on demand) or automatically on a scheduled basis.

By understanding and using the log files, you can:

Gather and Analyze

- Run-time performance data posted by a target (i.e., a policy enforcement point (PEP) or a run-time monitoring component such as Insight).
- Run-time event data (transaction events, policy violation events, etc.) posted by a target.
- Approval history data.
- Design/change-time policy data.
- Audit data.
- Federation job queue data.
- Consumer Registration data.

Troubleshoot Problems

For example, if your CentraSite needs to add more disk storage due to an increase in the number of assets, you can use Audit log files to see what percentage the asset activities has increased by and plan for the amount of new disk storage you need.

System-Defined CentraSite Logs

CentraSite can maintain the following types of system logs:

- **Policy Log.** The Policy Log contains information about the design/change-time policies that CentraSite has executed. By default, CentraSite only logs information about policies that fail. However, you can optionally configure CentraSite to log information about policies that resulted in success, informational, warning, and failure alerts.

In addition, if you purge the policy log, CentraSite makes an entry in the policy log to indicate that entries have been purged.

- **Approval History Log.** The Approval History Log contains a record of all approval requests that have been triggered by a policy with an approval workflow action. This log shows the status of each approval request that has been submitted to CentraSite.
- **Audit Log.** An Audit Log reports on the creation/update activities of a particular asset (including changes in an asset's lifecycle state).
- **Run-Time Event Log.** The Run-Time Event Log contains information about run-time events that have occurred in a target (i.e., a policy-enforcement point (PEP) or a run-time monitoring component).
- **Run-Time Performance Log.** The Run-Time Performance Log captures run-time metrics for all assets and publishes them to CentraSite at regular intervals.

Configuring Logs

CentraSite uses the system-defined log settings to store activities and performances of some of the registry objects, such as policies and assets, as log records. However, in some cases, you might want to modify the log settings configuration, in order to determine the optimal performance for CentraSite.

You can configure log settings by executing the following command in the command line interface `CentraSiteCommand.cmd` (Windows) or `CentraSiteCommand.sh` (UNIX) of Command Central. The tool is located in <*CentraSiteInstallDir*> /utilities folder.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

Prerequisites

To be able to configure the log settings, please note the following points:

- To configure from a command line, the CentraSite Registry Repository must be online, and the tool requires a Java 6 runtime.
- Changes to the log configuration do not affect the currently running tasks.

Configuring Log Settings from the Command Line

To invoke log setting from the command line, you must perform the following high-level steps:

1. Create a configuration (config.xml) file.
2. Execute the script file with appropriate input parameters.

Important: You must be a user with CentraSite Administrator role to execute the script file.

Creating a Configuration File to Define Log Setting

You can create a config.xml file that defines a log setting. Examine the config.xml file. It contains at least the XML namespace used for providing uniquely named elements and attributes.

Define a Log Unit

The unique element `LogUnit` represents the `com.centrasite.config.log.unit` property in config,xml. The `LogUnit` element specifies the log record to be stored in the CentraSite Log Database.

In the `LogUnit` element, add an attribute called `name` and assign it a value equal to either of the following:

Log Unit	Description
`Policy_Log`	Contains information about the design/change-time policies that CentraSite has executed.
`Approval_Log`	Contains information about all approval requests that has been triggered by a policy with an approval workflow action.
`Audit_Log`	Contains information on the creation/update activities of a particular asset (including changes in an asset's lifecycle state).
`Runtime_Event_Log`	Contains information about run-time events that have occurred in a target (i.e., a policy-enforcement point (PEP) or a run-time monitoring component).
`Runtime_Performance_Log`	Contains information about the KPI metrics of all rogue assets.

This `LogUnit` element should look similar to this:

```
<LogUnit name=Policy_Log></LogUnit>
```

Define Log Settings

The `com.centrasite.config.log.setting` property represents the `LogSetting` attribute. This attribute defines specific information about the performance of the registry object.

In this section, you create log settings that will be used within the log unit definition.

Within the `LogUnit` element, create `LogSetting` attributes say, `Success`, `Failure` etc.

You can configure the `LogUnit` element with one of the appropriate `LogSetting` attributes.

Log Unit	**Description**	
	Log Setting	**Logs**
Policy_Log	**Success**	Policies that have resulted in success alert.
	Info	Policies that have resulted in informational alert.
	Warning	Policies that have resulted in warning alert
	Failure	Policies that have resulted in failure alert.
	For more information, see "Policy Log" on page 348	
Approval_Log	**Log Setting**	**Logs**
	Log_Approval	Approval policies that have resulted in Approved state.
	For more information, see "Approval History Log" on page 349.	
Audit_Log	**Log Setting**	**Logs**
	Enable_Audit	All activities (such as creation, modified etc) performed on a particular asset.
	For more information, see "Audit Logs" on page 351.	
Runtime_Event_Log	**Log Setting**	**Logs**
	Policy_Violation_Events	Monitors and tracks all the policy violation events.

Log Unit	Description	
	Transaction_Events	Monitors and tracks all the transaction events.
	Monitoring_Events	Monitors and tracks all the monitoring events.
	Lifecycle_Events	Monitors and tracks all the lifecycle events.
	For more information, see "Run-Time Event Logs" on page 352.	
Runtime_Performance_Log	**Log Setting**	**Logs**
	Log_Performance_Events	Monitors and tracks all the performance events
	For more information, see "Run-Time Performance Logs" on page 352.	

This `LogSetting` attribute should look similar to this:

```
<LogUnit name=Policy_Log>
    <LogSetting>Success</LogSetting>
    <LogSetting>Failure</LogSetting>
</LogUnit>
```

The final config.xml should look similar to this:

```
<LogSettings xmlns="http://namespaces.centrasite.com/configurations/logs">
    <LogUnit name=Policy_Log>
        <LogSetting>Success</LogSetting>
    </LogUnit>
    <LogUnit name=Approval_Log>
        <LogSetting>Log_Approval</LogSetting>
    </LogUnit>
    <LogUnit name=Audit_Log>
        <LogSetting>Enable_Audit</LogSetting>
    </LogUnit>
    <LogUnit name=Runtime_Event_Log>
        <LogSetting>Transaction_Events</LogSetting>
        <LogSetting>Policy_Violation_Events</LogSetting>
        <LogSetting>Monitoring_Events</LogSetting>
        <LogSetting>Lifecycle_Events</LogSetting>
    </LogUnit>
    <LogUnit name=Runtime_Performance_Log>
        <LogSetting>Log_Performance_Events</LogSetting>
    </LogUnit>
</LogSettings>
```

If you do not specify a log setting value or if you specify an incorrect log unit or log setting value (for example, `Log Approval`), the command execution fails and a warning message is issued.

```
(this is correct)
<LogUnit name=Policy_Log>
  <LogSetting>Success</LogSetting>
</LogUnit>
(this is incorrect due to missing LogSetting element)
<LogUnit name=Policy_Log>
</LogUnit>
(this is incorrect due to invalid LogSetting value)
<LogUnit name=Policy_Log>
  <LogSetting>Success_and_Failure</LogSetting>
</LogUnit>
```

Note: Make sure you copy this file somewhere within the file system of the machine where CentraSite is installed.

Executing the Script File That Invokes Log Setting

To execute the script file that invokes log setting, use a command of the following format:

```
CentraSiteCommand set Log [-url <CENTRASITE-URL>] -user <USER-ID>
-password<PASSWORD> -file <CONFIG-FILE>
```

Example

```
CentraSiteCommand set Log -url http://localhost:53307/CentraSite/CentraSite
-user Administrator -password manage -file config.xml
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `set Log` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

Monitoring Logs

When you've configured CentraSite to log messages you can monitor its activities by viewing the log messages. By examining the log files you can monitor many aspects of CentraSite’s events.

Policy Log

The Policy Log contains information about the design/change-time policies that CentraSite has executed. By default, CentraSite only logs information about policies that fail. However, you can optionally configure CentraSite to log information about policies that resulted in success, informational, warning, and failure alerts. To view the policy log, you must belong to a role that includes the View Policy Log permission.

To view the policy log

1. In CentraSite Control, go to **Administration > Logs > Policy Log**.
2. Complete the following fields to specify which type of log entries you want to view:

In this field...	Specify...
Object Name	*Optional*. A pattern string that describes the names of the objects (of **Object Type**) whose log entries you want to view. You can provide the exact name or use a pattern string consisting of a character sequence and/or the % wildcard character (which represents any string of characters). For example, if you specify the pattern string ‘A%’, CentraSite displays entities whose names start with ‘A’. Leave **Entity Name** empty to view all names.
Policy Type	The type of policy whose log entries you want to view. To view the log entries for design/change-time policies, select **Design/Change Time** from the drop-down list (if it is not already selected).
Object Type	The object type whose log entries you want to view.
Event Type	The event type whose log entries you want to view.
Policy Status	The policy execution status that you want to view. A policy's execution status is the result set of each of its action's execution

In this field...	Specify...
	result. CentraSite writes the following policy execution status to the policy log depending on the log configuration:

Icon	Description
🟩	*Success*. Displays policies that have resulted in success alert.
ℹ	*Info*. Displays policies that have resulted in informational alert.
△	*Inprogress*. Displays policies that have resulted in inprogress alert.
⚠	*Warning*. Displays policies that have resulted in warning alert.
🔴	*Failure*. Displays policies that have resulted in failure alert.

In this field...	Specify...
Execution Date	*Optional*. The time period that you want to examine. Leave the **From** and **To** fields empty to view log entries for all dates.

3. Click **Search** to retrieve the specified log entries.
4. To view details for a particular entry in the returned list, click the name of the policy.

Note: If a policy included a WS-I action, the log entry for the policy will include a link to the results of the WS-I action.

Approval History Log

The Approval History Log contains a record of all approval requests that have been triggered by a policy with an approval workflow action. This log shows the status of each approval request that has been submitted to CentraSite. To view the Approval History Log, you must belong to a role that includes the `View Approval History` permission.

To view the Approval History Log

1. In CentraSite Control, go to **Administration > Logs > Approval History**.
2. Filter your search for approval log information by completing the following fields and clicking **Search**:

In this field...	Do the following...
Object Name	The object that was submitted for approval.
Approval Flow Name	The name assigned to the approval workflow.
Requestor	Click **Select User** and select the user who requested the approval.
Approver	Click **Select User(s)** and select the user(s) who approved the request.
Request Date From/To	Use the calendar to specify a date range for the requests.
Approval Date From/To	Use the calendar to specify a date range for the approvals.
Status	Select a status of the approval request (e.g., In Progress, Approved, etc.).

3. To view details of a particular approval workflow, click the hyperlinked value in the **Approval Flow Name** column.
4. The **Approval Flow Information** panel provides the following information about the approval workflow.

Field	Description
Approval Flow Name	Name of the approval workflow.
Mode	Mode of the approval workflow (e.g., Anyone or Everyone).
Status	Status of the approval policy (e.g., In Progress, Approved, New, No Action, Pending or Rejected).
Creation Date	Date when the approval workflow was created.

5. The **Requestor Summary** panel provides the following information about the requestor of the approval workflow.

Field	Description
Requestor Name	User who triggered the approval workflow.
Approval Type	Name of the approval action (for example, Initiate Approval or Initiate Group-dependent Approval)
Entity	Name of the entity on which the approval was triggered.
Reason for Request	Additional comments or descriptive information stating the reason for the approval request.

6. The **Approver Summary** panel provides the following information about the approver(s) of the approval workflow.

Field	Description
Approver	User who approved or rejected the approval workflow.
Action	Action of the approver (e.g., Approved or Rejected).
Comments	Additional comments or descriptive information stating the reason for approval or rejection.

Audit Logs

An audit log reports on the creation/update activities of a particular asset (including changes in an asset's lifecycle state). You can view the audit logs of any asset that you can view.

To view an asset's audit log

1. Display the asset's detail page and select the **Audit Log** profile.
2. The **Audit Log** profile displays the following information:

Field	Description
Event Type	The type of event that was executed on the asset (e.g., created or updated).
Date/Time	The date and time that the event was executed.

Field	Description
User	The user who executed the event.
Comment	Descriptive information about the event executed on the asset.

Run-Time Event Logs

The run-time event log contains information about run-time events that have occurred in a target (i.e., a policy-enforcement point (PEP) or a run-time monitoring component).

The target publishes to CentraSite the run-time events that have occurred (assuming that the target type contains a MIB file in its target type definition file). CentraSite provides predefined event types for use with webMethods Mediator or any third-party PEP that is integrated with CentraSite.

Each asset has its own run-time event log, which is located on the **Events** profile on its detail page. By default, all the predefined event types are logged, but you may disable any type.

You can view the run-time event log of any asset that you can view.

Users with the proper permissions can perform these additional tasks:

- View a log of all run-time events that have occurred in a particular target or in all targets system-wide.
- Create and manage custom run-time event types for use with webMethods Mediator or any third-party PEP that is integrated with CentraSite.

Run-Time Performance Logs

Targets capture run-time metrics for assets. If you are using the Mediator target, Mediator's data collector captures Key Performance Indicator (KPI) metrics for each virtual service and publishes them to CentraSite at regular intervals. If you are using a run-time monitoring component such as Insight, the monitoring component captures the KPI metrics of all rogue assets and publishes them to CentraSite at regular intervals.

Each asset has its own performance log, which is located on the **Performance** profile on its detail page. The Performance profile is similar for all assets.

You can view the performance log of any virtual service that you can view.

By default this log is enabled, but you may disable it.

Overview of Purging

The term obsolete refers to any data not needed by the database but still occupying space on it. This unwanted or obsolete data can eventually fill up the disk and decrease the database performance, causing time-consuming database lookups, contention issues and so on. The process of systematically removing this unwanted data from the database is called *purging*. This process basically involves running the Purger scripts. They are available for all log unitCentraSites stored in the CentraSite Log Database.

Typically, you need the CentraSite Log Database to include only the recent log records. For instance, each CentraSite object will have a number of actions performed on it, and for each action a separate audit log is generated and stored in the CentraSite Log Database. This will increase the size of the CentraSite Log Database dramatically. Failure to purge the excess log records in the database could cause problems. Therefore, the purge interval should be configured as required.

Each time you purge a log unit, the corresponding purging log entry is created and can be viewed in CentraSite Control. For example, when you purge a certain set of policy logs, the corresponding purging log entry is automatically created and will be visible in the Policy Log page.

Purging Logs

For performance reasons, CentraSite uses system log records to store activities and performances of some of the registry objects (say, policies, assets etc.) that are defined via the log settings. However, in some cases, you might want to modify the log settings configuration, in order to avoid performance degradation of CentraSite.

Prerequisites

To be able to configure the log purging, please note the following points:

- To configure from a command line, the CentraSite Registry Repository must be online, and the tool requires a Java 6 runtime.
- Changes to the log purging configuration do not affect the currently running tasks.

Note that CentraSite does not purge:

- Policy logs in "InProgress" state.
- Auditable events that log on object creation.
- Approval logs in "Pending" state.
- Federation job queue data.
- Consumer registration data.
- Purged audit log data (that is, records of the audit logs that were purged earlier).

Note: When you purge policy log entries, CentraSite creates a new informational policy log entry indicating that policy log entries have been purged. If any such informational policy log entries exist from previous invocations of the policy log purger, they are also purged, so that the only such policy log entry remaining is the one for the current invocation.

Configuring Log Purging from the Command Line

You can configure log purging by executing the following command in the command line interface `CentraSiteCommand.cmd` (Windows) or `CentraSiteCommand.sh` (UNIX) of Command Central. The tool is located in <*CentraSiteInstallDir*> /utilities folder.

If you start this command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

To invoke log purging from the command line, you must perform the following high-level steps:

1. Create a configuration (config.xml) file.
2. Execute the script file CentraSiteCommand with appropriate input parameters.

Important: You must be a user with CentraSite Administrator role to execute the script file.

Creating a Configuration File to Define Log Purging

You can create a config.xml file that defines a log purging. Examine the config,xml file. It contains at least the XML namespace used for providing uniquely named elements and attributes.

Define a Log Purge Setting

The `LogPurgeSetting` element specifies the type of log record to be purged from the CentraSite Log Database.

In the `LogPurgeSetting` element, add an attribute called `log` and assign it a value equal to either of the following:

Log Unit	Description
`Policy_Log`	Purge log record that contains information about the design/change-time policies that CentraSite has executed.

Log Unit	Description
`Approval_Log`	Purge log record that contains information about all approval requests that has been triggered by a policy with an approval workflow action.
`Audit_Log`	Purge log record that contains information on the creation/update activities of a particular asset (including changes in an asset's lifecycle state).
`Runtime_Event_Log`	Purge log record that contains information about run-time events that have occurred in a target (i.e., a policy-enforcement point (PEP) or a run-time monitoring component).
`Runtime_Performance_Log`	Purge log record that contains information about the KPI metrics of all rogue assets.

This `LogPurgeSetting` element should look similar to this:

```
<LogPurgeSetting log="Runtime_Event_Log">
</LogPurgeSetting>

<LogPurgeSetting log="Runtime_Performance_Log">
</LogPurgeSetting>
```

Define Log Purge Type

The log purge type attributes provide information about the different type of logs configured to be purged on different days, times and even how many days of logs to keep.

You can create the purge type attributes that will be used by the log purge setting definition.

Within the `LogPurgeSetting` element, create the following attributes (as required) and assign values.

Log Purge Setting	Description
`ExportLocation`	Specifies a location for the exported files on the database.
`Until`	Deletes log records and exports them as an archive to a configurable directory that can be imported later. However, if the export location is not specified, then CentraSite simply deletes the log records.

Log Purge Setting	Description
	Note: Specify the xs:dateTime values as required. The default time zone is UTC/GMT.
`OlderThan`	Deletes log records that are older than the specified date and exports them as an archive to a configurable directory that can be imported later. However, if the export location is not specified, then CentraSite simply deletes the log records. **Note:** Specify the xs:dateTime values as required. The default time zone is UTC/GMT.
`CommitThreshold`	Defines the threshold value for batch purging each of the log units. The default threshold value is `10000`.

The LogPurgeSetting element should now look similar to this:

```
<LogPurgeSetting log="Runtime_Event_Log">
     <OlderThan>-P5D</OlderThan>
</LogPurgeSetting>

<LogPurgeSetting log="Runtime_Performance_Log">
     <Until>2002-05-30</Until>
     <CommitThreshold>1000</CommitThreshold>
</LogPurgeSetting>
```

The final config.xml should look similar to this:

```
<LogPurgeSettings
  xmlns="http://namespaces.centrasite.com/configurations/logs">
    <LogPurgeSetting log="Approval_Log">
        <ExportLocation>/home/usr/admin/centrasite/log/backups/approval
        </ExportLocation>
    </LogPurgeSetting>
    <LogPurgeSetting log="Runtime_Event_Log">
        <OlderThan>-P5D</OlderThan>
    </LogPurgeSetting>
    <LogPurgeSetting log="Runtime_Performance_Log">
        <Until>2002-05-30</Until>
        <CommitThreshold>1000</CommitThreshold>
    </LogPurgeSetting>
</LogPurgeSettings>
```

Note: Make sure you copy this file somewhere within the file system of the machine where CentraSite is installed.

Executing the Script File That Invokes Log Purging

To invoke log purging function, you must use the script file CentraSiteCommand with the config.xml input parameter, as follows:

```
CentraSiteCommand purge Logs
```

```
[-url <CENTRASITE-URL>]
-user <USER-ID>
-password <PASSWORD>
[-assets <Asset
Name/Key (s)>]
-file <CONFIG-FILE>
```

Example

```
CentraSiteCommand purge Logs -url
"http://localhost:53307/CentraSite/CentraSite" -user "Administrator" -password
"manage" -assets "MyService" -file "config.xml"
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `purge Logs` utility:

Parameter	Description
`-url`	The fully qualified URL (http://localhost:53307/CentraSite/CentraSite) for the CentraSite registry/repository.
`-user`	The user ID of a user who has the CentraSite Administrator role.
`-password`	The password of the user identified by the parameter `-user`.
`-assets`	The name or UUID assigned to the asset and which uniquely identifies it within the registry.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

Note: If the specified key or name of the asset to purge does not match any existing asset in the registry, then no purging is done.

Exporting the Purged Log Records

You can selectively export log records to a location on disk.

Exporting the Purged Log Records from the Command Line

You can modify the config.xml file to specify the export location of purged log records and the date range to purge log records, and execute the command line utility to export the defined log records. For example, if you specify a directory location and the date until which the log records need to be purged, then executing the command line, CentraSite deletes the log records that were generated until the specified date and exports them as archive to the specified directory.

The following table identifies the log purging behavior of CentraSite for each purge setting configuration:

Log Purge Settings	Until		Older Than	
	Export Location Specified	Export Location Not Specified	Export Location Specified	Export Location Not Specified
Date Specified	Deletes log records that are generated until the specified date and exports them as an archive to the specified directory that can be imported later.	Permanently deletes log records that were generated until the specified date from the CentraSite Log Database.	Deletes log records that are older than the specified date and exports them as an archive to the specified directory that can be imported later.	Permanently deletes log records that are older than the specified date from the CentraSite Log Database.
Date Not Specified	Deletes log records that are generated until the current date and exports them as an archive to the specified directory that can be imported later.	Permanently deletes log records that are generated until the current date from the CentraSite Log Database.	Deletes log records until the current date and exports them as an archive to the specified directory that can be imported later.	Permanently deletes log records until the current date from the CentraSite Log Database.

Configuring the Purger Properties for High Volume Data Handling

Over time, a very large number of log records will accumulate in the log database, which makes the handling of log records difficult, and directly affects the performance of a log database. Therefore, it is important to purge unwanted log records from the log database occasionally.

However, purging large number of log records at one time would definitely result in system failures. In order to avoid such failures, CentraSite supports purging of the log records in a batch mode. Purging of log records in a batch mode can be initiated by enabling the `enable.partial.commit` property to `true` in the purger.properties file. The purger.properties file is located in the logpurging/resources folder under the CentraSite installation directory.

Note: Disabling the `enable.partial.commit=true` property would lead to failures if the system is not able to process the huge amount of log records.

Basically, the log records accumulate at a different rate for each log unit. CentraSite defines a default threshold value for batch purging each of the log units as follows:

```
##partial commit enabled/disabled
enable.partial.commit=true
##policy log threshold
policy.log.partial.commit.threshold=1000
##approval log threshold
approval.log.partial.commit.threshold=500
##events threshold
runtime.events.partial.commit.threshold=1000
##metrics log threshold
runtime.performance.log.partial.commit.threshold=1000
## No of days before the current day for which the data will not be included
## in the purging
## in case of Export Log Entries & Delete Log Entries
## (i.e. No date criteria specified).
## purge.older.than=1
```

You can modify the threshold values for batch purging the log units as required.

Note: If the purging type in the config.xml file is set to **Export** (No date criteria), then purging would exclude the log records by the number of days defined in the `purge.older.than` property. For example, if the purging is scheduled with `purge.older.than=7`, then log records that are available a ahead of the 7 days would be purged from the log database.

To change the batch purging properties

1. Edit the purger.properties that is located in the CentraSite installation directory.
2. Enable/disable batch purging setting the `enable.partial.commit` property to `true` or `false`, as required. The default value is "true".

3. Specify the threshold values in the `partial.commit.threshold` property of each log unit, as required.
4. Save and close the file.

   The changes will take effect immediately in the next purging.

Removing Leftover Auditable Events

In previous releases of CentraSite, some auditable events remained in the log after purging, even though the events referred to CentraSite objects that no longer existed.

There is one known situation where many of these events were created: the Integration Server can be configured to write metrics information for virtual services into CentraSite at regular intervals. Each metrics update resulted in some leftover events.

If you have upgraded your CentraSite Registry Repository from a previous release, such leftover auditable events might still exist in the log. To remove these auditable events, you can use a command line tool. The tool consists of an executable jar file in the bin folder of the CentraSite installation. It requires a Java 6 runtime and needs to be called in the following way:

```
java -jar CentraSiteOptimizeAuditableEvents.jar <DBURL> <administratorID> <password>
```

For example:

```
java -jar CentraSiteOptimizeAuditableEvents.jar
"http://localhost:53307/CentraSite/CentraSite" DOMAIN\admin pAsSw0rD
```

Note that you need the "CentraSite Administrator" role to run this tool.

10 Registry Federations in CentraSite

Overview of Registry Federations in CentraSite

CentraSite provides support for federated registries. Federation allows you to obtain copies of objects from one registry and store them in another registry, where you intend to access and navigate them.

The registries participating in a federation can be CentraSite registries and UDDI v3-compliant registries. Currently, the UDDI federation support allows federating data from a 3rd party UDDI registry such as jUDDI or SAP to CentraSite.

Overview of the Federation Process

The UDDI-based federation allows you to federate data between two UDDI v3 compliant registries. The data to be federated is inquired from the source UDDI registry using the UDDI Inquiry API. The obtained data is transformed into one or more XML-formatted events whose structure corresponds to an event type in the Event Type Store, and the events are emitted across the event bus. When an event reaches the UDDI endpoint, the data to be federated is extracted from the event and published to the target registry using the UDDI publish API.

The federation process consists of the following steps:

- Creating a federation configuration file that specifies the source and target registries and the objects you want to copy. This file is used as input for the commands to add and trigger the federation.
- Adding the federation configuration to the list of target registries defined for federation.
- Triggering the federation as required. This can be either on an ad hoc basis, or using the scheduling mechanism of your operating system.
- When a federation is no longer required, it can be removed from the list of target registries defined for federation.

You can carry out these steps using the command line script `CentraSiteCommand`, which is located in <*CentraSiteInstallDir*> /utilities.

Federation is carried out from a source UDDI registry to one or more target CentraSiteregistries. Typically, the source and target registries are on different machines, and they are addressed by their URL. The federation commands (create, add, trigger, etc.) must be run on a machine where CentraSiteis installed, i.e. on one of the target machines.

The list of target registries defined for federation is maintained in a file named emit.xml on the target machine where the federation commands are run. This file is located at <*CentraSiteInstallDir*> /Federation/config/templates/emit.UddiFederation/OSGI-INF/blueprint/.

When you add a federation, details of the federation are added to emit.xml, and when you trigger a federation, the information contained in emit.xml is used to route the federated data to the target registry. Thus, the commands for adding and triggering a federation must run on the same machine.

If there is more than one target registry, each machine that hosts a registry can in theory contain its own emit.xml file with its own list of target registries defined for federation. However, it is generally simpler to maintain an emit.xml file on just one of the target machines, and to trigger all of the federations as required from that machine.

Note that there is no automatic synchronization between the emit.xml files across multiple target registries; entries made in one emit.xml file are not copied automatically to other emit.xml files.

Creating a Federation Configuration File

To configure the UDDI federation, you need to supply the following details:

- The source registry details where the original data is located: host, port, username, password (needed for connecting to the registry for inquiring data).

  Also the URLs of the UDDI security and inquiry APIs, to allow the UDDI Client API to connect to the registry.
- The target registry details to which the data will be copied: host, port, username, password (needed for connecting to the registry for publishing data).

  Also the URLs of the UDDI security, inquiry and publish APIs, to allow the UDDI Client API to connect to the registry.
- The data that has to be inquired from the source registry. The types of UDDI object that can be federated are business entities, business services and tModels. Refer to the standard UDDI specification for details of these object types.
- The location of the Event Type Store. This is by default <*SuiteInstallDir*> /common/ EventTypeStore/.
- The location where further internal components required for the federation are available. This is by default <*SuiteInstallDir*> /CentraSite/Federation.

You can configure these details in an XML configuration file. A template of the XML configuration file is available in <*SuiteInstallDir*> /CentraSite/Federation/config/ federationConfiguration.xml. You can copy this file and edit it to suit your needs.

After configuring these details, you can start the federation.

Here is a sample configuration file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:federationConfiguration
xmlns:ns2="http://namespaces.softwareag.com/EDA/Federation/UDDI/Configuration"
xmlns:ns3="urn:uddi-org:api_v3">
 <sourceRegistryDetail>
  <host>MyUDDIServer</host>
  <port>8080</port>
  <user>root</user>
  <password>root</password>
  <securityUrl>http://MyUDDIServer:8080/juddiv3/services/security
     </securityUrl>
  <inquiryUrl>http://MyUDDIServer:8080/juddiv3/services/inquiry
     </inquiryUrl>
 </sourceRegistryDetail>

 <targetRegistryDetail>
  <host>FedTarget</host>
  <port>53307</port>
  <user>AdminUser</user>
  <password>ABCXYZ123</password>
  <defaultBusinessEntity>Default Organization</defaultBusinessEntity>
  <securityUrl>http://FedTarget:53307/UddiRegistry/security</securityUrl>
  <inquiryUrl>http://FedTarget:53307/UddiRegistry/inquiry</inquiryUrl>
  <publishUrl>http://FedTarget:53307/UddiRegistry/publish</publishUrl>
 </targetRegistryDetail>

 <inquiryParams>
  <find_business>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>Custom Organization</ns3:name>
  </find_business>

  <find_service>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>PlainService</ns3:name>
   <ns3:name>TestService</ns3:name>
  </find_service>

 <find_tModel>
   <ns3:findQualifiers>
    <ns3:findQualifier>caseInsensitiveMatch</ns3:findQualifier>
    <ns3:findQualifier>approximateMatch</ns3:findQualifier>
   </ns3:findQualifiers>
   <ns3:name>%</ns3:name>
  </find_tModel>
 </inquiryParams>

 <sourceBasicAuthentication>true</sourceBasicAuthentication>

 <targetBasicAuthentication>false</targetBasicAuthentication>

 <batchEventEmission>true</batchEventEmission>

 <batchPublish>false</batchPublish>

 <timestampBasedUpdate>true</timestampBasedUpdate>

 </ns2:federationConfiguration>
```

The source registry details are provided in the `sourceRegistryDetail` element.

The target registry details are provided in the `targetRegistryDetail` element. When the CentraSite registry is the target registry, the port number is by default 53307.

The securityUrl and inquiryUrl elements provide the location of the Security and Inquiry APIs that are required for accessing the UDDI registries. The publishUrl element for the target registry provides the location of the Publish API that is required for publishing the UDDI objects.

The securityUrl, inquiryUrl and publishUrl are optional for CentraSite target registries since these are the default settings.

If the configuration file does not contain the user and password details for the source and target registries, you will be prompted to supply these values when you initialize the federation.

The target registry has another configuration:

```
<defaultBusinessEntity> Default Organization </defaultBusinessEntity>
```

This option is to configure the default business Entity (Organization) in the target registry. All business services will be associated with a particular business entity in the source registry as shown here:

```
<businessService
  serviceKey="uddi:juddi.apache.org:68f6379d-9063-4eaf-b35a-45199c9f0b0a"
businessKey="uddi:juddi.apache.org:4375a871-b13e-46ae-b329-c79871efc3f1">
```

The business key refers to a business entity in the source registry. When you try to publish this service to some other registry, it could be the case that the business entity to which this service is associated with may be already available in the target, in which case this service can be published as is in the target registry. If the target does not have the business entity to which the service is associated, you might still want to publish the service separately from the entity. The defaultBusinessEntity element provides the option to provide the default organization in the target registry to which the service must be associated when it is published. If the business key is present in the target, then the service is published without changes, and if the business key is not present, then the business key of the organization referred to in the XML file is associated with the service and it is published. In the example here, the Default Organization refers to the organization (business entity) in CentraSite.

Use the inquiryParams element in the configuration file to specify the objects that you want to select for federation.

Within this element, you can specify the following child elements:

- find_business: search for UDDI objects of type business entity.
- find_service: search for UDDI objects of type service.
- find_tModel: search for UDDI objects of type tModel.

These are the UDDIv3 Inquiry API calls. Their syntax must adhere to the UDDIv3 standard.

Not all of these child elements need to be supplied, so for example if you want to federate only business entities, it is sufficient to provide the find_business element.

The name element within the find_business, find_service or find_tModel element specifies the name of the object or objects to be searched for. You can use wildcards in the `name` element to search for multiple objects; these are the standard wildcards as described in the UDDI specification. For `find_business` and `find_service` you can supply more than one `name` element.

You can restrict the set of selected objects by using a filter, specified in a `findQualifier` element. The filter modifies the way in which the value in the `name` element is interpreted. The available filters are the findQualifiers as described in the UDDI specification. The example above illustrates how to use the findQualifiers to use case sensitivity/insensitivity, and exact/approximate match.

Basic Authentication

The `sourceBasicAuthentication` and the `targetBasicAuthentication` elements define whether the authentication to the UDDI registry should be via authTokens or through HTTP Basic Authentication. They take Boolean values of either true or false.

Authentication to a UDDI registry usually works through the Security API of the UDDI standard through authTokens. The messages to UDDI are sent as SOAP messages over HTTP. Some UDDI registries provide an option to authenticate via HTTP Basic Authentication. CentraSite and jUDDI, for example, provide authentication via authTokens whereas SAP registry provides authentication via HTTP Basic authentication.

So the `sourceBasicAuthentication` element specifies this option for the source registry. If set to true, it means that the source will be authenticated via HTTP basic authentication and if set to false, the source will be authenticated via authTokens. The `targetBasicAuthentication` element configures the authentication method for the target registry. If set to true, authentication is done via HTTP Basic authentication and if set to false, the authentication is done via authTokens.

Note: Currently, the federation from an SAP source registry to a target CentraSite registry only supports HTTP Basic Authentication on the source SAP registry, i.e. authenticating via AuthToken on the source SAP registry is not supported. Therefore, the value in the `<sourceBasicAuthentication>` element must be `true`. Similarly, if SAP is the target registry, then the value in the `<targetBasicAuthentication>` element must be `true`.

Batch Event Emission

You can control whether the UDDI objects found by the inquiry calls should be sent individually to the target or as a batch containing all of the found objects. If you set `batchEventEmission` to true, the objects will be sent as a batch. If you set the value to false, the objects will be sent individually.

For example, if the inquiry calls return 10 UDDI objects for federation (these 10 objects could be businesses, services, tModels or mixture of all), and `batchEventEmission` is true, all 10 UDDI objects are emitted as one single XML event. If `batchEventEmission` is false, each of the 10 UDDI objects will be emitted individually.

Batch Publish

This configuration defines how the UDDI objects should be published. This is in alignment with the behavior of the UDDI Publish API.

If `batchPublish` is set to true, then events that were batched due to the setting of `batchEventEmission` will also be published on the target machine as a batch. If an error occurs while publishing one of the batch's objects, this object will not be published and all of the remaining objects in the batch will be ignored. This is the standard behavior of the UDDI Publish API.

If `batchPublish` is set to false, then events that were batched due to the setting of `batchEventEmission` will nevertheless be published individually on the target machine. If an error occurs while publishing one of the batch's objects, this object will not be published, but the processing of the remaining objects in the batch will continue.

Details of the publish operation will be provided in a log file.

Timestamp Based Update

This is an optimization configuration. During a federation operation, it can happen that an object being federated is already present in the target machine due to a previous federation operation. The following situations can occur:

- The object that is already available in the target registry might have been updated in the source registry and now this object has to be updated in the target registry as well.
- The object that is already available in the target registry has not been updated in the source registry, so the objects are identical.

Before an object is published is the target machine, CentraSite checks if the object already exists on the target machine. If the object already exists, but the timestamps are different, then the object is updated in the target registry. If the object to be published is identical to the object already present, the object is not updated. This results in performance optimization, thereby reducing the number of publish operations to the target registry.

If `timestampBasedUpdate` is set to true, then CentraSite performs the comparison and updates the target object if needed. If `timestampBasedUpdate` is set to false, the comparison is not performed and the object is published, regardless of whether the objects are updated objects or not.

Adding a UDDI Federation to the Federation List

After you have created the configuration file for a UDDI federation, you can make the federation available for triggering by adding it to the list of target registries defined for federation. To do this, use a command of the following form:

```
CentraSiteCommand add Federation [-user <USER-ID>] [-password <PASSWORD>]
-name <NAME> -file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `add Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the target machine who has the permissions to create objects on the target machine.
`-password`	The password of the user identified by the parameter `-user`.
`-name`	A name for the federation.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

For example:

```
CentraSiteCommand add Federation -user AdminUser -password ABCXYZ123
-name FedName -file C:\Federation1.xml
```

If the configuration file contains the user and password information in the `targetRegistryDetail`*NAME*, you can omit these from the command line. If you supply the user and password values on the command line and also in the configuration file, the values on the command line will be used.

Every federation has a dedicated name that you can provide in the *Name* parameter. This name is displayed when you use the `list Federation` command to show the list of target registries defined for federation. It is also required when you wish to remove a federation using the `remove Federation` command.

When the federation has been added to the federation list, you can trigger the federation at any time.

When you add a UDDI federation as described above, CentraSite creates an entry for the federation in the emit.xml file.

Triggering the UDDI Federation

When you trigger a federation, you activate the federation of objects from a source UDDI registry to one or more target registries, as defined in the federation's configuration file. If you wish to run the federation on a regular basis, for example once per hour or once daily, you can use the scheduling mechanism provided by the operating system to trigger the federation at the appropriate time intervals.

The federation has to be triggered from the machine where the CentraSite instance is located. Hence, with CentraSite as the target registry, the federation will be triggered from the target machine.

To trigger a federation, use a command of the following form:

```
CentraSiteCommand trigger Federation [-url <CENTRASITE-URL>] [-user <USER-ID>]
[-password <PASSWORD>] -file <CONFIG-FILE>
```

The following table describes the complete set of input parameters that you can use with the `trigger Federation` utility:

Parameter	Description
`-url`	The URL of the CentraSite registry. This is needed for connection to the Federation service running in CentraSite. The default value is `http://localhost:53307/` if this parameter is not provided.
`-user`	The user ID of a user on the source machine who has the permissions to access the required objects on the source machine.
`-password`	The password of the user identified by the parameter `-user`.
`-file`	The absolute or relative path to the XML configuration file. If relative, the path should be relative to the location from where the command is executed.

For example:

```
CentraSiteCommand trigger Federation -user AdminUser
-password ABCXYZ123 -file C:\Federation1.xml
```

If the configuration file contains the user and password information in the `sourceRegistryDetail`, you can omit these from the command line. If you supply the user and password values on the command line and also in the configuration file, the values on the command line will be used.

On Windows, you can use the Task Scheduler to trigger the federation at regular intervals. The Task Scheduler is available under Start > All Programs > Accessories > System Tools > Task Scheduler. In the Task Scheduler, supply the following values:

- The command to be activated, for example:

  ```
  C:\SoftwareAG\CentraSite\utilities\CentraSiteCommand.cmd
  ```

  If the path contains blank characters, you need to enclose the value in double quotes.

- The parameters of the command, for example

  ```
  trigger Federation -user AdminUser -password ABCXYZ123 -file C:\Federation1.xml
  ```

- The times when the command should be activated.

On Linux and UNIX, you can use the operating system's `cron` facility to trigger the federation at regular intervals.

Removing a UDDI Federation from the Federation List

You can remove a federation from the list of target registries defined for federation. To do this, use a command of the following form:

```
CentraSiteCommand remove Federation [-user <USER-ID>]
[-password <PASSWORD>] -name <Name>
```

The following table describes the complete set of input parameters that you can use with the `remove Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the source machine who has the permissions to access the required objects on the source machine.
`-password`	The password of the user identified by the parameter `-user`.
`-name`	The name of the federation.

For example:

```
CentraSiteCommand remove Federation -user AdminUser -password
ABCXYZ123 -name FedName
```

This command removes the federation defined by *Name* from the list of target registries defined for federation. The *Name* is that which you assigned when you added the federation to the federation list.

To run this command, you require permission at the operating system level to modify emit.xml.

Displaying the List of Target Registries Defined for Federation

You can display the list of IDs of target registries defined for federation by using a command of the following form:

```
CentraSiteCommand list Federation [-user <USER-ID>]
[-password <PASSWORD>] [-name <Name>]
```

The following table describes the complete set of input parameters that you can use with the `list Federation` utility:

Parameter	Description
`-user`	The user ID of a user on the target machine who has the permissions to create objects on the target machine.
`-password`	The password of the user identified by the parameter `-user`.
`-name`	The name of the federation to be displayed. If you do not supply this parameter, the names of all defined federations are displayed.

For example:

```
CentraSiteCommand list Federation -user AdminUser -password
ABCXYZ123 -name FedName
```

To run this command, you require permission at the operating system level to read emit.xml.

Mapping UDDI Object IDs and tModels

You can maintain uniqueness of UDDI object IDs during the federation process, and map UDDI structures on to JAXR-based structures in the target registry.

Key Generator Processor

Each object in a UDDI registry has a unique ID. When an object is federated into a target registry, its ID still needs to be unique, i.e. it must not be a duplicate of an existing object in the target registry. The federation process allows you define your own keys for objects federated into CentraSite. Such keys are called *publisher assigned keys*.

The use of publisher assigned keys is activated in the federation process by using a so-called *key generator* that assigns key name domains to users who are publishers. This means that when a UDDI object is added to the CentraSite registry by a given user, the user-specific domain name is added to the UDDI object ID.

The use of key generators and publisher assigned keys is known from the UDDI standard.

The relationship between a user and the assigned domain name is maintained in a <process> element in the federation configuration file. This element has attributes that specify:

- name of user/publisher
- domain name

Example: The ID of an object in a UDDI registry is uddi:abcd1234-5678-9abc-def0-123456789abc. The domain associated with the user is `mydomain`. When the object is federated to CentraSite, the object ID will be uddi:mydomain:abcd1234-5678-9abc-def0-123456789abc.

Adding a Key Generator Processor for Federation

To add a key generator processor for federation, you have to modify the emit.xml file as follows:

To add a key generator processor for federation

1. Configure the details of the target registry in the configuration file for which you want to add the key generator processor, as described in "Creating a Federation Configuration File" on page 363. Execute the `add Federation` command and provide a unique identifier for this target. This will configure the target for federation and add an entry for the federation route in emit.xml.
2. Now open the emit.xml file. It will have the targets configured for federation. Locate the `<pipeline>` element that contains the attribute `id="<unique_identifier>"`, the identifier that you provided in the previous step while executing the `add Federation` command.
3. Include a `<process>` element immediately after `<pipeline id="..." >` but before `<to uri="..." />` as shown:

   ```
   <pipeline id="<federationId>" ... >
      <process ref="KeyGeneratorProcessor_<federationId>" />
      <to uri="uddi://_<federationId>" ... />
   </pipeline>
   ```

4. It is extremely important that you replace *<federationId>* with the unique identifier that you provided while executing the command. After you add the `<process>` element, you now have to add the following details after the `<camelContext>` element as shown:

   ```
   <camelContext>
   . . . .
   </camelContext>

   <bean id="KeyGeneratorProcessor_<federationId>"
          class="com.softwareag.centrasite.federation.
          uddi.component.UddiKeyGeneratorProcessor" >
      <property name="domainKey" value="<domainKey>" />
      <property name="federationId" value="<federationId>" />
   </bean>
   ```

5. The bean ID that you provide and the `ref` value in <process ref="..."> must be the same. The class name must not be modified and must be the same as provided in this example. Add the two properties `<domainKey>` for providing the domain name for the key generator and *<federationId>* to identify the federation route.
6. Now execute `add Federation` with the same identifier and the same target details as before. The newly added processor has to be configured for federation and this will accomplish it.

Removing a Key Generator Processor

You might wish to remove the key generator processor at a later stage. To do this, proceed as follows:

To remove a key generator processor

1. Remove the `<process>` element and the `<bean>` element.
2. Execute the `add Federation` command after this so that the processor will not be used when you trigger the federation.

If the key generator processor and the taxonomy processor are both defined, the key generator processor runs first, then the taxonomy processor runs.

Taxonomy Processor

Objects in a UDDI registry are classified according to concepts within tModels. In a registry that supports JAXR, such as CentraSite, objects are classified according to categories within taxonomies. The mapping from a UDDI classification to a JAXR-based classification uses the key value and tModel name that are contained in the `keyedReference` element of the UDDI object. The tModel name is used as the taxonomy name, and the key value is used as the category name. If the taxonomy or category do not already exist in the target registry, they are created automatically.

The concept and classification scheme are stored in tModel and keyed references of the tModel in CategoryBags of UDDI objects. The taxonomy processor processes the keyedReferences from categoryBags of the UDDI objects and creates the concepts and classification schemes in the target registry.

For example, the following are two `<keyedReference>` elements from a tModel's `<categoryBag>`:

```
<keyedReference keyValue="sap.com/LMINTERNALAGENT"
   keyName=""
   tModelKey="uddi:uddi.sap.com:categorization:software-component"/>

<keyedReference keyValue="I12"
   keyName=""
   tModelKey="uddi:uddi.sap.com:categorization:business-system"/>
```

The tModelKey in the keyedReference points to a tModel. The taxonomy processor checks if this tModel can be mapped to a taxonomy in JAXR. This check is done on the basis of the UDDI to JAXR-based mapping standard. Please refer to the JAXR standard for information on how this mapping works. If the tModel can be identified as a taxonomy in JAXR according to the mapping standard, then the tModel corresponding to the tModelKey is published as a taxonomy (ClassificationScheme) in the target registry. The keyValue corresponds to concepts (or categories) in the taxonomy. So the above keyedReferences will create the following taxonomy/concept hierarchy in the target registry:

```
uddi-sap:softwareComponent
|
+-- sap.com
```

```
    |
    +-- LMINTERNALAGENT
uddi-sap:businessSystem
|
+-- I12
```

Adding a Taxonomy Processor

To add a taxonomy processor, you have to edit the emit.xml file and add a few details. Follow the instructions as given below:

To add a taxonomy processor

1. Configure the details of the target registry in the configuration file for which you want to add the taxonomy processor, as described in "Creating a Federation Configuration File" on page 363. Execute the `add Federation` command and provide a unique identifier for this target. This will configure the target for federation.

2. Now open the emit.xml file from the above mentioned location. It will have the targets configured for federation. Locate the `<pipeline>` element with the attribute id="<*unique_identifier*>", the identifier that you provided in the previous step while executing the `add Federation` command.

3. Include a <process> element immediately after `<pipeline id="..." ...>` but before `<to uri="..." />` as shown:

   ```
   <pipeline id="<federationId>" . . . . >
      <process ref=" TaxonomyProcessor_<federationId>" />
      <to uri="uddi://_<federationId>" . . . . />
   </pipeline>
   ```

4. In case you want to configure both the key generator and the taxonomy processor, you have to provide the key generator `<process>` element first and the taxonomy processor `<process>` element next as shown:

   ```
   <pipeline id="<federationId>" ... >
      <process ref=" KeyGeneratorProcessor_<federationId>" />
      <process ref=" TaxonomyProcessor_<federationId>" />
      <to uri="uddi://_<federationId>" ... />
   </pipeline>
   ```

5. It is extremely important that you replace <*federationId*> with the unique identifier that you provided while executing the command. After you add the `<process>` element, you now have to add the following details after the `<camelContext>` element as shown:

   ```
   <camelContext>
   . . . .
   </camelContext>

   <bean id="TaxonomyProcessor_<federationId>"
            class= "com.softwareag.centrasite.federation.uddi.component.
            UddiTaxonomyProcessor">
     <property name="csUri" value="<csUri>" />
     <!-- Example: http://<host>:<port>/CentraSite/CentraSite -->
     <property name="federationId" value="<federationId>" />
   </bean>
   ```

6. The bean ID that you provide and the `ref` value in <process ref="..."> must be the same. The `class=` name must not be modified and must be the same as provided in this example.

Add the two properties `csUri` for providing the Registry URL for the processor and `federationId` to identify the federation route.

7. Now execute `add Federation` with the same identifier and the same target details as before. This configures the newly added processor for federation.

Removing a Taxonomy Processor

You might wish to remove the taxonomy processor at a later stage. To do this, proceed as follows:

To remove a taxonomy processor

1. To remove the taxonomy processor, remove the `<process>` element and the `<bean>` element.
2. Execute the `add Federation` command after this so that the processor will not be used when you trigger the federation.

If the key generator processor and the taxonomy processor are both defined, the key generator processor runs first, then the taxonomy processor runs.

Users and Access Rights

The federation can be triggered by the a user who has the permissions needed for the source and the target registries. The target registry user specified in the configuration file must have the required permission to publish the data on the target registry, otherwise the publish operation will fail. For example, if CentraSite is the target registry, then the target registry user requires the Asset Provider role permission in order to publish the data.

On the target registry, the federated data will be owned by the specified target registry user. For example, if CentraSite is the target registry, and the target registry user is given as usersec1, then usersec1 will be the owner of the federated data on the target registry.

Users other than the owner can access the data on the target registry if they have the required privileges. For example, if CentraSite is the target registry, then a user having the Asset Consumer role permission can access the data.

11 Suite Usage Aspects

Overview of Suite Usage Aspects

Products of the webMethods suite use the CentraSite Registry Repository for storing and maintaining their objects and object types. These objects and object types should normally only be maintained by the suite products themselves, rather than by a CentraSite user who has access to such objects and object types. Thus, for example, a user of CentraSite Control who has access to the objects and object types, such as a user with the CentraSite Administrator role, must avoid making any change that would cause the objects or objects types to become unusable by the suite products.

Versioning Assets

CentraSite offers a versioning capability, which allows you to maintain several versions of an asset. This feature is described in the *CentraSite User's Guide* and *Working with the CentraSite Business UI*.

When you generate a new version of an asset, CentraSite adds a new asset of the same type to the catalog. The new asset has the same name and description as the one from which it was versioned, and has an updated version number. The new version is related to the old version by a `Supersedes` association from the new version to the old version. In cases where the detail page of an asset has a Summary profile, the association is displayed under the Summary profile.

The versioning capability for the asset types defined by the suite products is by default not activated. Unless the documentation for the suite components states otherwise, do not activate the versioning for these asset types.

Modifying or Deleting Assets

CentraSite users with the appropriate permissions can delete existing assets. This standard functionality is described in *CentraSite User's Guide* and *Working with the CentraSite Business UI*.

Generally, asset instances that are created and used by suite products are governed by policies that prevent their modification or deletion. The predefined design/change-time policy Prevent Editing of webMethods Assets is an example of such a policy. This means that a CentraSite user will normally not be able to modify or delete the asset, due to the rules defined in the policy.

However, a user with the Modify Assets permission, or with a permission that implies the Modify Assets permission, can modify assets regardless of the policy settings. Also, if the suite product does not define a deletion policy for its assets, this gives CentraSite users with the appropriate permissions the theoretical possibility of modifying or deleting the assets. If you modify or delete such an asset in CentraSite, this may lead to inconsistencies or errors in the product that created the asset, if that product requires the asset definition to be still present and unchanged.

Deletion of an asset that belongs to a suite asset type should normally only be done using the Retract feature of the suite's Designer user interface, rather than as a CentraSite user.

Publishing an Asset

When a suite product tries to update an asset that it previously created in CentraSite, the update will fail if the asset is locked due to a pending activity at the CentraSite level. This can happen, for example, if a CentraSite policy related to the asset has been triggered that requires an approval action, and the approval has not yet been granted.

Example: A user publishes a package from the Designer UI and this publish creates a service among other things in CentraSite. Then a CentraSite user with appropriate permissions tries to change the lifecycle state of this service and this triggers a policy you have defined for the service. If this policy has an approval action, then as long as the approval is pending no-one can edit this asset. This is as per design and is the same for all asset types (both suite and otherwise). If the user now tries to publish the package again, the publishing logic will try to edit the service and will fail since CentraSite will not allow edits. The policy can be triggered by a CentraSite user as well. Basically if you have a policy for the specific asset type with approval action, you may run into this situation.

Communication with Designer UI

Integration Server assets can be published at the package level to CentraSite from the Package Navigator view of the Designer. For publishing assets to CentraSite, the publisher uses CentraSite connection information provided in the Integration Server administration console.

The CentraSite connection information must specify a valid CentraSite URL and a user name which the publisher uses for authentication purposes to communicate with CentraSite.

If you change the CentraSite connection information in any way, for example, if you delete the user from CentraSite, it will not be possible to publish any assets from the Integration Server to CentraSite.

Remote Connection from Designer

Connecting remotely to CentraSite from Designer may result in time-outs depending on the individual environment and due to subsequent changes of the IP address (TCP/IP stack, VPN, etc.).

In such cases it may be necessary to restart/reconnect the following components:

- Software AG Runtime
- CentraSite Registry Repository
- Any client involved

User Accounts

Suite products generally communicate with CentraSite using existing CentraSite user accounts. Therefore, CentraSite administrators must be aware that any modification of the scope of these user accounts (e.g. changing the password or permissions of the user) could affect the functionality of the suite product.

Before you modify the definition of a user account, check if any connected suite products use the user account, and ensure that the modification will not limit the functionality of the suite products.

UDDI Clients

If you are using a UDDI client to access CentraSite, note the following points:

- The UDDI term *Business* is the equivalent of the CentraSite term *Organization*. If you store a UDDI Business object in CentraSite, it is stored internally in CentraSite as an Organization object and will be visible in CentraSite Control as an Organization object. When you retrieve the object via UDDI, it will of course be returned as a Business object.
- If you are using UDDI V2 to communicate with CentraSite, note that the predefined CentraSite UDDIv2 Inquiry Policy is by default not active. This policy is required for UDDI V2 processing.

  To activate the policy, open CentraSite Control, display the list of design/change-time policies, select the UDDIv2 Inquiry Policy and set its lifecycle state to `Productive`.

Using CentraSite with ARIS

Profile for ARIS Properties

The asset type Process contains a profile named **ARIS Properties** which includes attributes that are of use when CentraSite is integrated with the ARIS products. The contained attributes are:

The **ARIS Properties** profile includes the following attributes:

Attribute	Description
ARIS GUID	The Global Unique Identifier (GUID) that is assigned to the asset and uniquely identifies it once imported into ARIS.
ARIS Client	Download URL for ARIS Web client.
ARIS Publisher URL	URL for assets published through ARIS Publisher.
ARIS Connect URL	URL to establish connection to the ARIS Connect.

The asset type Process contains the ARIS Properties profile by default. An administrator can optionally choose to disable this profile in the asset type definition. For example, if ARIS is not jointly used with CentraSite, an administrator might want to disable this profile for all assets on type Process.

Configuring CentraSite for Use with ARIS

ARIS uses CentraSite to retrieve/publish services and processes for process execution. When users define business processes with ARIS Architect, they can optionally publish the resulting processes and related services to CentraSite.

CentraSite includes predefined lifecycle models and predefined design/change-time policies to facilitate the governance of the business processes that ARIS publishes.

Note: When users working on the same ARIS database are created as users in different CentraSite organizations, by default they will not have visibility to each others' assets. It is recommended to either have all users of an ARIS database in the same CentraSite organization, or if this is not possible, to create a group in CentraSite containing all those users, and assign View or Modify permissions to the group.

Lifecycle Models That ARIS Uses

When ARIS publishes a business process to CentraSite, the business process is represented in the registry by a Process object. ARIS requires business processes (Process objects) to be governed by a lifecycle model. CentraSite includes a predefined lifecycle model called *BPM Process Lifecycle* for this purpose.

BPM Process Lifecycle Model

The *BPM Process Lifecycle* model defines the following states and transitions:

State	Has valid transitions to...
Requested	Implementing, Rejected
Implementing	Implemented
Implemented	Testing
Testing	Tested
Tested	Available
Available	Retired
Retired	*This is an end state. It has no outgoing transitions.*
Rejected	*This is an end state. It has no outgoing transitions.*

Certain operations performed in ARIS trigger policies in CentraSite, and these policies change the lifecycle state of a given Process object. When a user deletes a business process in ARIS, a policy in CentraSite automatically switches that process to the Retired state in the registry.

When you install CentraSite, this predefined lifecycle model is disabled (i.e., inactive). To use CentraSite with ARIS, you must activate the lifecycle model, as described in "Activating the Lifecycle Models and Design/Change-Time Policies That ARIS Uses" on page 387.

Service Lifecycle Model

The *Service Lifecycle* model defines the following states and transitions:

State	Has valid transitions to...
Proposed	In Design, Rejected, Retired
In Design	In Development, Rejected
In Development	Implemented
Implemented	In Test
In Test	Tested, In Development

State	Has valid transitions to...
Tested	In Production
In Production	Under Maintenance, Deprecated
Under Maintenance	In Production
Deprecated	In Production, Retired
Retired	*This is an end state. It has no outgoing transitions.*
Rejected	*This is an end state. It has no outgoing transitions.*

Certain operations performed in ARIS trigger policies in CentraSite, and these policies change the lifecycle state of a given Service object. When a user deletes a service in ARIS, a policy in CentraSite automatically switches that process to the Retired state in the registry.

When you install CentraSite, this predefined lifecycle model is disabled (i.e., inactive). To use CentraSite with ARIS, you must activate the lifecycle model, as described in "Activating the Lifecycle Models and Design/Change-Time Policies That ARIS Uses" on page 387.

Customizing the Predefined Lifecycle Model for Process Objects

If the predefined lifecycle models that CentraSite provides does not suit your needs, you can customize them or create a new model for Process objects or Service objects. If you do this, however, you must also update the predefined policies that switch the lifecycle state of objects in the registry. For more information about which policies change the state of an object, see "Design/Change-Time Policies That ARIS Uses" on page 383.

Note: The CentraSite Community Edition does not support the use of customized lifecycle models for Process objects or Service objects. If you are using the Community Edition, you must use the predefined lifecycle model that CentraSite provides.

Design/Change-Time Policies That ARIS Uses

CentraSite includes the following predefined policies.

Policy Name	Purpose
ARIS Delete Processes	Switches a specified Process object to the Retired lifecycle state. This policy executes when a process model is deleted by a Delete Process call from ARIS. Note that if the Process object in CentraSite is not in a state that permits a transition to the Retired state (i.e, if a transition to the Retired state is not permitted by the Process object's lifecycle model) the policy will fail. **Important:** If you customize the predefined lifecycle model that CentraSite installs for Process objects, or if you apply a different lifecycle model to Process objects, you must modify this policy so that it switches the Process object to the appropriate state in your customized lifecycle model.
ARIS Delete Services	Switches a specified Service (native or virtual) to the Retired lifecycle state. This policy executes when a service is deleted by a Delete Service call from ARIS. Note that if the Service in CentraSite is not in a state that permits a transition to the Retired state (i.e, if a transition to the Retired state is not permitted by the Service's lifecycle model) the policy will fail. **Important:** If you customize the predefined lifecycle model that CentraSite installs for Services, or if you apply a different lifecycle model to Services, you must modify this policy so that it switches the Service to the appropriate state in your customized lifecycle model.
ARIS Release Processes	Switches a specified Process object to the Available lifecycle state. This policy executes when a process model is released by a Release Process call from ARIS. Note that if the Process object in CentraSite is not in a state that permits a transition to the Available state (i.e, the transition to the Available state is not permitted by the Process object's lifecycle model), the policy will fail. **Important:** If you customize the predefined lifecycle model that CentraSite installs for Process objects, or if you apply a different lifecycle model to Process objects, you must modify this policy so that it switches the Process object to the appropriate state in your customized lifecycle model.

Policy Name	Purpose
ARIS webMethods Integration	Sets instance-level permission and profile-level permissions to specified users/group for a Process object in CentraSite. ■ This policy contains the action Set Instance and Profile Permissions, which assigns the instance-level permission and optionally profile-level permissions to a specific set of users/groups (called the ARIS webMethods Integration Group) for the Process object and related assets. The ARIS webMethods Integration Group includes the group of users who can be availed in ARIS or Designer for ARIS webMethods Integration. ■ You can apply this policy when the following events occur to a Process object: ■ PostCreate ■ PreUpdate ■ OnTrigger
Enforce Unique Name	Ensures that the names of Application Server objects, IS Server objects and IS Connection objects that are created in CentraSite are unique. ■ This policy contains the action Enforce Unique Name, which you can configure to: ■ Enforce the unique name requirement for Application Server objects, IS Server objects and IS Connection objects in all organizations defined in CentraSite. ■ Allow different versions of an object to exist in CentraSite with same the name. ■ You can apply this policy when the following events occur to an object: ■ PreCreate ■ PreUpdate
Notify ARIS on Service Changes	Notifies the ARIS APG service endpoint when a Service object (native or virtual) in CentraSite is updated. ■ This policy contains the action Notify ARIS Service, which notifies the ARIS APG Service endpoint with the SOAP request message provided in this action. The APG Service endpoint is picked up from the associated ARIS Application Server.

Policy Name	Purpose
	■ You can apply this policy when the following events occur to a Service object: ■ PostUpdate ■ PostStateChange ■ OnTrigger
Notify ARIS on Service Completion	Notifies the ARIS APG service endpoint when a Service object (native or virtual) in CentraSite when a user changes the state of the service to a "completed" lifecycle state (e.g, the Productive state). ■ This policy contains the action Notify ARIS Service, which notifies the ARIS APG Service endpoint with the SOAP request message provided in this action. The APG Service endpoint is picked up from the associated ARIS Application Server. ■ You can apply this policy when the following events occur to a Service object: ■ PostStateChange ■ OnTrigger
Notify ARIS on Service Deletion	Notifies the ARIS APG service endpoint when a Service object (native or virtual) in CentraSite is deleted. ■ This policy contains the action Notify ARIS Service, which notifies the ARIS APG Service endpoint with the SOAP request message provided in this action. The APG Service endpoint is picked up from the associated ARIS Application Server. ■ You can apply this policy when the PostDelete event occurs to a Service object.
Reset Lifecycle State to Initial State	Resets the lifecycle state of a Process object if it is republished from ARIS. ■ This policy contains the action Set State, which you can set to any state of the BPM Process Lifecycle model. ■ You can apply this policy when the OnTrigger event occurs to a Process object.

Activating the Lifecycle Models and Design/Change-Time Policies That ARIS Uses

You use the following procedures to activate the predefined lifecycle models (LCM) and design/change-time policies used by ARIS.

When you install CentraSite, the lifecycle models and design/change-time policies are in the inactive state.

Note: The activation of the predefined lifecycle models (LCM) and design/change-time policies used by ARIS is not supported if you are using a CentraSite Community Edition license.

Activating the ARIS-Related LCMs and Policies

Use the following procedure to activate the lifecycle models and design/change-time policies for ARIS.

To activate the ARIS-related LCM and policies

1. Open CentraSite Control.
2. Activate the ARIS lifecycle models using the following steps:
   a. Go to **Administration** > **Lifecycles** > **Models**.
   b. Locate the lifecycle model called "BPM Process Lifecycle" and activate it.
   c. Locate the lifecycle model called "Service Lifecycle" and activate it.
3. Activate the state-change policies using the following steps:
   a. Go to **Policies** > **Design/Change Time** to display the policy list.
   b. Enable the **Show Predefined Policies** option to display the predefined policies that CentraSite provides.
   c. Activate each of the following policies using the procedure in the *CentraSite User's Guide*:
      - ARIS Release Processes
      - ARIS Delete Processes
      - ARIS Delete Services
      - ARIS webMethods Integration
      - Reset Lifecycle State to Initial State

   For more information about these policies, see "Design/Change-Time Policies That ARIS Uses" on page 383.

4. Configure and activate each of the following policies using the procedure described in "Configuring the ARIS Change-Notification Policies" on page 388:
   - Notify ARIS on Service Changes
   - Notify ARIS on Service Completion
   - Notify ARIS on Service Deletion

   For more information about these policies, see "Design/Change-Time Policies That ARIS Uses" on page 383.
5. Configure and activate the Enforce Unique Name using the procedure described in "Configuring the Enforce Unique Name Policy" on page 390.

   For more information about this policy, see "Design/Change-Time Policies That ARIS Uses" on page 383.
6. Configure and activate the Reset Lifecycle State to Initial State using the procedure described in "Configuring the Reset Lifecycle State to Initial State Policy" on page 391

   For more information about this policy, see "Design/Change-Time Policies That ARIS Uses" on page 383.

Configuring the ARIS Change-Notification Policies

Use the following procedure to configure the Notify ARIS Service action in the ARIS change-notification policies (i.e., the Notify ARIS on Service Changes, Notify ARIS on Service Completion and Notify ARIS on Service Deletion policies). The change-notification policies inform the ARIS server of changes that have been made to objects in the registry that belong to ARIS.

To configure a change-notification policy

1. Go to **Policies** > **Design/Change Time** to display the policy list.
2. Enable the **Show Predefined Policies** option to display the predefined policies that CentraSite provides.
3. Click the policy that you want to configure.
4. Click the Notify ARIS Service action on the **Actions** tab to open the Edit Action Parameters page, and then do the following.

   a. Set the following parameters:

Parameter	Description
`HTTP Basic Auth Enabled`	*Boolean* Specifies whether the service is secured by Basic HTTP authentication. If you enable this option, you can optionally specify the user ID and password that CentraSite is to submit when it invokes the service in the following parameters. If you leave these parameters empty, CentraSite will submit

Parameter	Description
	the credentials belonging to the user who triggered this policy action. `HTTP Basic Auth Username` — The user ID that you want CentraSite to submit for HTTP basic authentication (if you do not want CentraSite to submit the user ID of the user who triggered the policy). `HTTP Basic Auth Password` — The password associated with the user ID specified in `HTTP Basic Auth Username`.
`SOAP Request Message`	*String* The SOAP message that CentraSite is to submit to the ARIS service. The default message (shown below) includes substitution tokens that insert run-time data into the message. The substitution tokens are described in the documentation for the `Send Email Notification` action in the *CentraSite User's Guide*. You can use these substitution tokens to customize this message.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.idsscheer.com/age/webMethods/">
   <soapenv:Header/>
   <soapenv:Body>
      <web:UpdateServiceRequest>
           <dbname>${context.ARIS_DB_CONTEXT}</dbname>
           <language>${user.locale}</language>
           <serviceDetail>
             <guid>${entity.key}</guid>
             <name>${entity.name}</name>
             <url>${entity.URL}</url>
             <lifeCycleState>${entity.state}
             </lifeCycleState>
             <owner>${entity.owner}</owner>
             <description>${entity.description}
             </description>
             <organization>${entity.organization}
             </organization>
             <version>${entity.version}</version>
             ${entity.attribute.Operations}
           </serviceDetail>
      </web:UpdateServiceRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

Parameter	Description
`SOAP Action`	*String* The SOAP action that CentraSite will set in the message. If you do not set this parameter, CentraSite will set the SOAP action to the empty string.

Parameter	Description
`Connection Timeout (in milliseconds)`	*Number* The length of time in milliseconds that CentraSite will wait for a response from the remote machine. If the timeout limit is exceeded, the policy action fails.
`Content Type`	*String* The value that CentraSite is to assign to the Content-Type header in the SOAP request that it submits to the service. Default: `text/xml`.

b. Click **Save** to save the parameter settings.

5. Click **Save** to save the updated policy.
6. Activate the policy. For more information, see the *CentraSite User's Guide*.

Configuring the Enforce Unique Name Policy

Use the following procedure to configure the `Enforce Unique Name` action in the `Enforce Unique Name` policy. This policy ensures that the names of the Application Server type objects that are created in CentraSite are unique.

To configure the Enforce Unique Name policy

1. Go to **Policies** > **Design/Change Time** to display the policy list.
2. Enable the **Show Predefined Policies** option to display the predefined policies that CentraSite provides.
3. Click the `Enforce Unique Name` policy.
4. Click the `Enforce Unique Name` action on the **Actions** tab to open the Edit Action Parameters page, and then do the following:

   a. Set the following parameters:

Parameter	Description
`Enforce Across Organizations`	*Boolean* If this parameter is set to True, then the unique name requirement for Application Server type objects is enforced in all organizations defined in CentraSite.
`Allow Different Versions`	*Boolean* If this parameter is set to True, then different versions of an Application Server type object can exist in CentraSite with same the name.

The event scope of this action is:

- PreCreate

- PreUpdate

b. Click **Save** to save the parameter settings.

5. Click **Save** to save the updated policy.
6. Activate the policy. For more information, see the *CentraSite User's Guide*.

Configuring the Reset Lifecycle State to Initial State Policy

Use the following procedure to configure the `Set State` action in the Reset Lifecycle State to Initial State policy. This policy resets the lifecycle state of a Process object if it is republished from ARIS.

To configure the Reset Lifecycle State to Initial State policy

1. Go to **Policies** > **Design/Change Time** to display the policy list.
2. Enable the **Show Predefined Policies** option to display the predefined policies that CentraSite provides.
3. Click the Reset Lifecycle State to Initial State policy.
4. Click the Set State action on the **Actions** tab to open the Edit Action Parameters page, and then do the following.

a. Set the following parameter:

Parameter	Description
`Change State To`	*String* Set the parameter to any state of the "BPM Process Lifecycle" model.

The event scope of this action is:

- OnTrigger

b. Click **Save** to save the parameter setting.

5. Click **Save** to save the updated policy.
6. Activate the policy. For more information, see the *CentraSite User's Guide*.

Activating the ARIS-Related LCMs and Policies in CentraSite

In CentraSite, the ActivatePoliciesForARIS command-line utility can be used for:

- Activate the lifecycle models and design/change-time policies that ARIS requires.
- Configure the Notify ARIS Service action in the ARIS change-notification policies (i.e., the Notify ARIS on Service Changes, Notify ARIS on Service Completion and Notify ARIS on Service Deletion policies). The change-notification policies inform the ARIS server of changes that have been made to objects in the registry that belong to ARIS.

- Configure the Enforce Unique Name action in the Enforce Unique Name policy. This policy ensures that the names of the Application Server type objects that are created in CentraSite are unique.
- Configure the Set State action in the Reset Lifecycle State to Initial State policy. This policy resets the lifecycle state of a Process object if it is republished from ARIS.

This is of a great usage for the Community edition of CentraSite as policies cannot be manipulated via its user interface.

Note: To run the ActivatePoliciesForARIS utility, you must either have access to the command line on the machine where the CentraSite Application Server Tier is installed or that system must be accessible via HTTP. Additionally, you must belong to the CentraSite Administrator role or a role that has permission to manage system-wide lifecycle models and system-wide design/change-time policies.

Activate Policies for ARIS from a Java Class

Use the following procedure to activate ARIS-related LCMs and Policies from a Java class:

To run the ActivatePoliciesForARIS command-line utility

1. Create a script file. For more information about writing a script file, see "Creating a Script File for Activating the ARIS LCM and Policies" on page 394.
2. Execute the script file with the following parameters:

   ```
   yourScriptFile -user yourCSUserID -password yourPassword
   [-url yourCSURL | [-h yourHostname -p yourPort]]
   ```

   Example

   ```
   myScript -user jcallen -password j45Hk19a [-url
   http://myserver:53305/CentraSite/CentraSite | [- h myserver -p 53305]]
   ```

   Note: If the registry/repository component is not running on the same machine as the Software AG Runtime, or is running on a port other than 53305, you must also include the -url parameter on the command line, followed by the URL for the CentraSite registry/repository (e.g., -url http://myserver:53305/CentraSite/CentraSite).

Activate Policies for ARIS from a Command Line

You can activate policies for ARIS using the command line tool `EnableARISIntegration.cmd` (Windows) or `EnableARISIntegration.sh` (UNIX). The command line tool is located in <*SuiteInstallDir*>/CentraSite/utilities.

If you start the command line tool with no parameters, you receive a help text summarizing the required input parameters.

The parameters of the command are case-sensitive, so for example the parameter `-url` must be specified as shown and not as `-URL`.

If you omit the passwords from the command, you will be prompted to provide them.

To activate policies for ARIS, use a command `EnableARISIntegration` of the following format:

On Windows

```
EnableARISIntegration.cmd -user <USERNAME> -password <PASSWORD>
[-dburl <CENTRASITE-URL> | [-h <HOSTNAME> -p <PORT>]]
```

On UNIX

```
EnableARISIntegration.sh -user <USERNAME> -password <PASSWORD>
[-dburl <CENTRASITE-URL> | [-h <HOSTNAME> -p <PORT>]]
```

Input Parameters

The following table describes the complete set of input parameters that you can use with the `EnableARISIntegration` utility:

Parameter	Description
USERNAME	*Required*. Your CentraSite user ID.
PASSWORD	*Required*. The password for your CentraSite user account.
CENTRASITE-URL	The fully qualified URL for the CentraSite registry/ repository. If you omit this parameter, the importer assumes that the registry/repository resides at `http://localhost:53305/CentraSite/CentraSite`. **Note:** If the registry/repository is running on a different machine and port number, you can use this parameter to specify its location instead of using the individual -h and -p parameters. (If you specify the -dburl parameter with the -h and/or -p parameters, the -h and -p parameters will be ignored.)
HOSTNAME	The host name or IP address of the computer where the CentraSite registry/repository component is running. If you omit this parameter, the importer assumes that the registry/repository is running on `localhost`.
PORT	The port number on which the CentraSite registry/ repository is configured to listen for incoming requests.

Parameter	Description
	If you omit this parameter, the importer assumes that the registry/repository is listening on the default port, `53305`.

Creating a Script File for Activating the ARIS LCM and Policies

The ActivatePoliciesForARIS utility is a Java class whose main() method executes when you execute ActivatePoliciesForARIS from the command line. To ensure that the CLASSPATH and other environment variables are set properly when you execute this utility, you must create a script file that calls ActivatePoliciesForARIS as described below.

Creating a Script File for Windows (a .bat File)

Create a script file that looks as follows, if CentraSite is running under Windows.

```
@echo off
set JAVAEXE=fullPathToJava.exe set REDIST=CentraSiteHomeDirectory\redist
set ARISPATH=CentraSiteHomeDirectory\rts\aris
set BASEDIR=%~dp0
cd /d %REDIST%

REM build CLASSPATH with all files from jar directory
set LOCAL_CLASSPATH=
for %%I in (".\*.jar") do call
    "CentraSiteHomeDirectory\bin\cfg\lcp.cmd" %%I

set BASEDIR=%~dp0
cd /d %ARISPATH%
set ARIS_CLASSPATH=
for %%I in ("%ARISPATH%\*.jar") do call
"CentraSiteHomeDirectory\bin\cfg\lcp.cmd" %%I

set LOCAL_CLASSPATH=%LOCAL_CLASSPATH%;%ARIS_CLASSPATH%
cd /d %REDIST%

%JAVAEXE%
   -cp %LOCAL_CLASSPATH% com.centrasite.aris.util.ActivatePoliciesForARIS %*
cd /d %BASEDIR%
```

Example

```
@echo off
REM
REM Run ARIS LCM and Policy Activator
REM
set JAVAEXE=D:\Fix7\CentraSiteGE7\jdk1.5.0_12\bin\java
set REDIST=C:\SoftwareAG\CentraSite\redist
set ARISPATH=C:\SoftwareAG\CentraSite\rts\aris
set BASEDIR=%~dp0
cd /d %REDIST%

REM build CLASSPATH with all files from jar directory
set LOCAL_CLASSPATH=
for %%I in (".\*.jar")
   do call "C:\SoftwareAG\CentraSite\bin\cfg\lcp.cmd" %%I
```

```
set BASEDIR=%~dp0
cd /d %ARISPATH%
set ARIS_CLASSPATH=
for %%I in ("%ARISPATH%\*.jar") do call
"C:\SoftwareAG\CentraSite\bin\cfg\lcp.cmd" %%I

set LOCAL_CLASSPATH=%LOCAL_CLASSPATH%;%ARIS_CLASSPATH%
cd /d %REDIST%

%JAVAEXE% -cp      %LOCAL_CLASSPATH%
    com.centrasite.aris.util.ActivatePoliciesForARIS %*
cd /d %BASEDIR%
```

Creating a Script File for Unix (C-Shell Script)

Create a script file that looks as follows if CentraSite is running under Unix.

```
set javaexe="fullPathToJava.exe"
set redist="CentraSiteHomeDirectory/redist"
set arispath="CentraSiteHomeDirectory/aris"
set mainjar="CentraSiteARISIntegration.jar"
set delim='\:'
cd "$redist"
set cl=""
foreach j ( `ls *.jar` )
  if ($cl != "") set cl=${cl}${delim}
  set cl=${cl}${j}
end
set delim='\:'
cd "$arispath"
set acl=""
foreach j ( `ls *.jar` )
  if ($acl != "")
  set acl=${acl}${delim}
set acl=${acl}${j}
end
setenv CLASSPATH ${mainjar}${delim}${cl}${acl}
cd "$redist"
$javaexe com.centrasite.aris.util.ActivatePoliciesForARIS $*
```

Example

```
#!/bin/csh
#
# Run ARIS LCM and Policy Activator
#
set javaexe="/.../softwareag/cjp/v16/bin/java"
set redist="/.../softwareag/CentraSite/redist"
set arispath="/.../softwareag/CentraSite/rts/aris"
set mainjar="CentraSiteARISIntegration.jar"  set delim='\:'
# build CLASSPATH with all files from jar directory
cd "$redist"
set cl=""
foreach j ( `ls *.jar` )
  if ($cl != "") set cl=${cl}${delim}
  set cl=${cl}${j}
end
set delim='\:'
cd "$arispath"
set acl=""
foreach j ( `ls *.jar` )
  if ($acl != "") set acl=${acl}${delim}
  set acl=${acl}${j}
```

```
end
setenv CLASSPATH ${mainjar}${delim}${cl}${acl}
cd "$redist"
$javaexe com.centrasite.aris.util.ActivatePoliciesForARIS $*
```

Provisioning CentraSite Services into ARIS Architect

CentraSite provides Web service operations for provisioning the Integration Server (IS) services and Web services into ARIS Architect.

The Web service operations filter Integration Server (IS) Services that are classified under the concept called WMAssetType ->ReusableAsset. An Integration Server service is classified under this concept, when the **Reuse** property value is set to public in Designer. This helps in filtering the default Integration Server services (example **wMPublic**), if they are published to CentraSite.

Following are the list of Web service operations that help in provisioning CentraSite services into ARIS:

- findAllServices
- getAllAssociatedServices
- getAllServiceDetails

These operations are part of SearchService (Web service).

Each of these operations have their object types differentiated using a property element that comes as part of the response payload as below.

Object Type Representation for Integration Server Service

```
<ns5:property><name>ServiceType</name><value>IS Service</value></ns5:property>
```

Object Type Representation for Web Service

```
<ns5:property><name>ServiceType</name><value>Service</value></ns5:property>
```