Bluenog Platform BluenogCMS Repository Import-Export Guide

Document Title:	Bluenog Platform BluenogCMS Repository Guide
Document Version:	3.1.9sp1
Effective Date:	March 31 2008

Import/Export Utility

The generic usage syntax for the import/export utility is as follows:

```
java -jar bluenog-content-tools-<version>-uber.jar <command> <command
properties> <namespace properties>
```

The <command> can be either *import* or *export*. The command properties is the properties filename and location relative to where you're running the jar file. These properties are relevant to the processing of the command. The namespace properties is the filename and location of the properties file that has the repository namespace mappings.

Importing into the Repository

The general principle behind the bulk import of documents and associated properties is that they are laid out on the filesystem in the hierarchy you would like to appear in your repository. Please take care to plan out this hierarchy in advance to suit your organization's needs. After the directory structure is laid out, the properties associated with a particular directory are specified in a file named: *<name of directory>.properties*. The file belongs in the parent directory of the directory in question. Similarly, the properties for the documents to be imported are the name of the document with *".properties"* appended to it. They belong in the same directory as the document itself. Documents are to be in a well-formed XML format. If no properties are specified, a minimal set of default properties will be set by the repository on import.

Here is a sample hierarchy:

```
import-root
   |
    --- repo
        repo.properties
         |
          --- content
              content.properties
               |
                --- document.xml
                    document.xml.properties
```

The configuration for the import is relatively straightforward. Here is a sample configuration:

```
path.separator=/
source.location.root=/path/to/some/import
source.location.path=

// The destination location. Path can either be a folder or a file.
destination.location.host=localhost
destination.location.port=60000
destination.location.root=/default
destination.location.path=

// authentication of the destination location
destination.authentication.username=root
destination.authentication.password=password
```

The key parameters to change are the location of your documents and hierarchy on your local filesystem; *source.location.root* and the location of the repository you're importing to via the *destination.location* properties.

Exporting an Existing Repository

To export your documents and hierarchy onto your local filesystem, from an existing repository, there are a few additional configurations to take into consideration. Here is a sample configuration properties file:

```
# The source location. Path can either be a folder or a file.

location.host=localhost
location.port=60000
location.rootpath=/default
location.path=

# authentication of the source location
authentication.username=root
authentication.password=password
plugin.1.classname=com.bluenog.tools.content.RepositoryExport

# The destination location. Path can either be a folder or a file.
plugin.1.configuration.destination.location.host=
plugin.1.configuration.destination.location.port=
plugin.1.configuration.destination.location.rootpath=/path/to/export
plugin.1.configuration.destination.location.path=


# authentication of the destination location
plugin.1.configuration.destination.authentication.username=root
plugin.1.configuration.destination.authentication.password=password

# namespace properties file
plugin.1.configuration.namespace.file=namespace.properties

# Properties file that contains properties to exclude from export
plugin.1.configuration.property.exclude.file=exclude.properties
```

The usage is similar to the import except now destination and source are reversed, the destination being a location on your local filesystem. There are two additional properties of note. One identifies the name and location of the *namespace properties*. This is the mapping of the fully qualified namespaces to an abbreviation to help the readability of the exported properties files. This is the same namespace properties files that is passed in when running the tool. Typically the namespace mapping configuration does not need to be modified.

Finally, there is a property that defines the name and location of a properties file that contains a listing of properties that you want to exclude from the export. Here is an example of an exclusion list:

```
# Properties to be excluded from the export

# Example:

exclude.property.1=H:UID

exclude.property.2=H:longdescription
```

Additional properties to be excluded can be added by simply increasing the number at the end of the key and providing the abbreviated namespace concatenated with the property name as the value.