


# Upgrading from 2.x

Upgrade utility not currently compatible with version 3.4.2

 The upgrader utility is not currently compatible with version 3.4.2 of Fedora, due to the Java package renaming in this release.

This issue is currently being addressed, see [the JIRA issue](#). We will announce on the mailing lists when a new version of the utility is available. In the meantime, you must upgrade to Fedora 3.3 as described below, then follow the instructions on [Upgrading from 3.x](#).

## Overview

This document explains how to migrate your Fedora repository from version 2.x to 3.x. Although it is written assuming 3.3, the same instructions apply if you are upgrading to an older 3.x release. Before continuing, you should familiarize yourself with the new [Content Model Architecture](#). A basic understanding of how the CMA works will be helpful in understanding the migration process.

Throughout this guide, `OLD_FEDORA_HOME` refers to the home directory of your Fedora 2.x installation, and `NEW_FEDORA_HOME` refers to the home directory of your new Fedora 3.x installation.

NOTE: To reduce confusion during this process, if you have previously ingested the Fedora demo objects, you should purge them from your 2.x repository before starting with the migration process. The demo objects have changed significantly since 2.x, and if you'd like the new versions, we recommend that you ingest them after performing a successful migration of your old repository.

## 1. Install, Start, and Stop Fedora 3.x

Follow the instructions in the Fedora [Installation and Configuration Guide](#). When finished, start the server and verify your installation was successful by visiting the `describeRepository` page (e.g. <http://localhost:8080/fedora/describe>). Finally, shut down Fedora. It should not be restarted again until later in the migration process.

## 2. Point Fedora 3.x to Existing Objects


### A. Object XML (FOXML)

Determine where your old object XML is stored. If you're unsure, consult `OLD_FEDORA_HOME/server/config/fedora.fcfg` and find the value of `object_store_base`.

Edit `NEW_FEDORA_HOME/server/fedora.fcfg` and change the value of `object_store_base` the *full path* of your old object's directory.

NOTE: The upgrade process will transform the objects in `object_store_base` of `NEW_FEDORA_HOME/server/fedora.fcfg` in place. Thus, by following the above instructions, the new upgraded files will exist in their old location. If that is not what is desired, then you could copy the old files (both objects and content) to their final location, point the `NEW_FEDORA_HOME/server/fedora.fcfg` to that new location, and then continue the update process.

Warning

 If you haven't already done so, make a backup of all original object XML files. These will be changed during the transformation part of the migration process, and if something goes wrong, this backup will be your only way of recovering.

### B. Managed Datastreams

Determine where your old Fedora Repository's managed Datastreams are stored. Again, you can check your old `fedora.fcfg` file. Look for the value of `datastream_store_base`.

Edit `NEW_FEDORA_HOME/server/fedora.fcfg` and change the value of `datastream_store_base` the *full path* of your old Datastreams directory.

NOTE: The migration process will not make any changes to these files.

## 2. Install the Migration Utilities

The migration utilities come as a separate download from the Fedora installation. They are all included, with source, in a single download [fedora-migration-3.2.zip at sourceforge.net](#). Once unzipped, you will find the executable jars in the top directory.

Although the filename is `fedora-migration-3.2.zip`, the utilities will also work for migrating to Fedora 3.3

## 3. Run the Analyzer

The analyzer examines all of your existing digital objects and looks for similarities. It outputs the following information in a directory you specify.

- A set of generated content models.
- A list of PIDs for each content model.
- Service Deployment (formerly known as Behavior Mechanism) information for each content model.

The analyzer does not make changes to any source objects.

## A. Configuring the Analyzer

The analyzer accepts a Java properties file for configuration.

Tip



In properties files, the "\" character must be escaped. When using Windows, this means paths like `c:\work\abc` must be written with two backslashes as a path delimiter rather than one.

Create a file (e.g. `migration.properties`) with the following content, filling in values appropriate to your environment:

```

# This is the directory where the analyzer's output files should be sent.
# If it doesn't already exist, it will be automatically created.
# If it already exists, it must be empty (to avoid accidental overwrites)
# To disable the above restriction, uncomment clearOutputDir=true below.
outputDir=c:\\fedora-migration
#clearOutputDir=true

# The Fedora 3.x home directory.
fedoraHome=c:\\fedora-3.3

# The full path to the JDBC driver Fedora is configured to use.
# NOTE: The analyzer only uses the database to aid in looking up
# the location of FOXML. It will populate the initial paths in
# the database the first time it runs, if the necessary.
jdbcJar=c:\\fedora-3.3\\tomcat\\webapps\\fedora\\WEB-INF\\lib\\postgresql-8.3-603.jdbc3.jar
# Aspects of the original objects to ignore for the purpose of
# classification. This is optional. If unspecified, the generated
# content models will be the MOST SPECIFIC POSSIBLE. If
# specified, this property must consist of a space-delimited
# list of any of the following:
# OrigContentModel
# If specified, objects that have differing values in the original
# contentModel property may be assigned to the same content
# model if they are otherwise similar.
# DatastreamIDs
# If specified, only datastreams bound to disseminators will
# be considered important for classification. Objects that have
# differing sets of UNUSED datastream IDs may be assigned to
# the same content model if they are otherwise similar.
# MIMETypes
# If specified, the MIMETYPE of each candidate datastream
# will be ignored for the purpose of classification. Objects that
# have differing MIME types for the same datastream may be
# assigned to the same content model if they are otherwise
# similar.
# FormatURIs
# This works exactly the same as MIMETypes, but applies to
# the FORMAT_URI of candidate datastreams.
#ignoreAspects=OrigContentModel DatastreamIDs MIMETypes FormatURIs

# Specific datastream IDs to ignore for the purpose of classification.
# This is optional. If specified, this property must consist of a
# space-delimited list of datastream IDs to ignore. Note: This configuration
# has no effect if DatastreamIDs is already specified as an ignoreAspect
# above.
ignoreDatastreamIDs=DC RELS-EXT RELS-INT POLICY

# Explicitly declare objects to be in the FedoraObject-3.0 content model.
# The default is 'false', or implicit. If left implicit, the objects
# will not have an explicit basic model, and it will be up to the
# system to use a default value at runtime. This option may have
# an impact on future upgrades. Future versions of Fedora may adopt a new
# basic model that has additional system methods, or require certain
# datastreams or formats. Objects that explicitly declare a 3.0
# model should behave exactly the same if Fedora is upgraded to a
# post 3.0 version. If left implicit, the objects may be interpreted
# in light of the new model, and may inherit new methods, or may
# fail validation and require updating if the new model introduces
# requirements they do not fulfill.

# Uncomment to force explicit basic model declarations in the
# upgraded objects.
# explicitBasicModel = true

```

## B. Analyzer Usage

The analyzer utility is an executable jar and takes the configuration file as a parameter. For example, if your configuration is in the current directory and is named migration.properties, enter the following:

```
java -jar analyzer.jar migration.properties
```

Analysis of small repositories will finish very quickly. For repositories with millions of objects, analysis will take several hours. With modern hardware, expect a rate of about 10,000 objects per minute.

#### Information

If upgrading to Fedora 3.3 with an embedded Derby database (a configuration that is not recommended for production), you may notice an error similar to "java.sql.SQLException: Cannot close a connection". This does NOT indicate that the analyzer failed, and can be safely ignored.

## C. Reviewing Analyzer Output

The analyzer will produce several files in the output directory.

Generated content models will be in FOXML 1.1 format and will be named `cmodel-n.xml` (where `n` is a number used for association). For each of these, there will be an associated `cmodel-n.members.txt` file containing a list of PIDs that conform to the content model.

Each content model object will contain the following inline XML datastreams:

- **CLASS-DESCRIPTION** - This is a simple human-readable log of the matching aspects of all member objects found during the analysis process. This datastream is not used or recognized by Fedora; it is only included for documentation purposes and may be removed at any time.
- **DS-COMPOSITE-MODEL** - This is a special Fedora-defined description of the datastreams (and aspects thereof) that member objects are expected to have. You'll notice that it contains a subset of the information expressed in CLASS-DESCRIPTION. This datastream is important, and should not be removed from the content model object.

If the member objects had disseminators:

- You will notice that the generated content model also has a `RELS-EXT` Datastream. This RDF Datastream points to the original Behavior Definition (now known as Service Definition) objects via a Fedora-defined `fedora-model:hasService` relationship. This relationship means that members of this content model should have the behaviors defined within the target SDef(s).
- There will also be an associated `cmodel-n-deployments.txt` file in the output directory. For each Behavior Mechanism formerly used by the objects' disseminators, this file identifies the original BMech, specifies a PID for a new, similar Service Deployment, and specifies a set of Datastream input name changes that the copy should have. This information is used by the generator to create new content-model-specific Service Deployment objects.

Three additional files will be created in the output directory:

- **sdefs.txt** - This lists all original Behavior Definition (now known as Service Definition) objects. The generator will create a stylesheet to upgrade these objects to FOXML 1.1.
- **sdeps.txt** - This lists all original Behavior Mechanism objects. Although these objects will be made obsolete by the new generated Service Deployments, the generator will create a stylesheet to upgrade them to Service Deployments in FOXML 1.1 so that you can view them in the Fedora 3.0 Repository before deciding to purge them.
- **nocmodel.txt** - This lists objects that should NOT be assigned a content model. Initially, the file is empty but it can be customized, if desired (see [Upgrading Objects Without Content Models](#) below).

## D. Customizing Content Model PIDs

By default, the PIDs assigned to the generated content models are of the form, `changeme:CModelN`. You should change these to use your own repository's namespace. You may also want to change the identifier part of the PID (e.g. `myns:Journal`).

To do this, open each `cmodel-n.xml` file in an editor and change the following as desired:

- The value of the PID attribute at the root of the document.
- If there is a `RELS-EXT` Datastream, change the part after `info:fedora/` in the `rdf:about` attribute. For example, change `rdf:about="info:fedora/changeme:CModel1"` to `rdf:about="info:fedora/myorg:Journal"`.

## E. Customizing Service Deployment PIDs

If any `cmodel-n-deployments.txt` files were created by the analyzer, you should also change any `NEW_DEPLOYMENT` pid values specified within each before continuing. These PIDs will default to values like `changeme:CModel1-BMech1`, but again, you should specify your own (e.g. `myns:Journal-DefaultImp`).

#### Warning

DO NOT specify the same PID for `NEW_DEPLOYMENT` as `OLD_BMECH`, as this will cause ingest problems later.

## F. Upgrading Objects Without Content Models

By default, the analyzer ensures that every data object is assigned to a content model. If you'd rather avoid assigning an explicit content model to an object, you may do so by a) removing its PID from the `cmodel-n.members.txt` file in which it resides and b) adding it to the `nocmodel.txt` file. This may be done for any number of objects.

#### Warning

In order for migration to work properly, the PID of every object in the source repository must occur exactly once in the set of PID list files.

## 5. Run the Generator

The generator reads the output of the analyzer (along with any customizations you have made) and adds the following to the same directory:

- New Service Deployment objects (as specified in each `cmodel-n-deployments.txt` file)
- Stylesheets for transforming existing objects as necessary



The generator does not make changes to the source objects.

### A. Configuring the Generator

Like the analyzer, the generator accepts a Java properties file for configuration.

The configuration file should have the following content, with values filled in appropriate to your environment.

Tip



Since you already entered `fedoraHome` and `jdbcJar` in `migration.properties`, you can minimize typing by just using that file, and adding the necessary `sourceDir` parameter as shown below.

```
# This is the directory containing the analyzer's output, and where the generator's
# results should be written.
sourceDir=c:\\fedora-migration

# The Fedora 3.x home directory
fedoraHome=c:\\fedora-3.3

# The full path to the JDBC driver Fedora is configured to use
jdbcJar=c:\\fedora-3.3\\tomcat\\webapps\\fedora\\WEB-INF\\lib\\postgresql-8.3-603.jdbc3.jar
```

### B. Generator Usage

The generator utility also accepts the configuration file as a parameter. For example:

```
java -jar generator.jar migration.properties
```

The generator should finish very quickly, regardless of the size of the repository.

### C. Reviewing Generator Output

The generator will produce several files in the output directory. These files, along with those already produced by the analyzer, will be used in the next steps of the migration process.

One stylesheet will be written for each PID list: `sdeps.xslt`, `sdefs.xslt`, `nocmodel.xslt`, and each `cmodel-n.members.xslt` file. Each of these stylesheets will include transformation rules for upgrading the objects to FOXML 1.1 and adding the necessary `fedora-model:hasModel` relationship via `RELS-EXT`, if necessary.

Information



The generated stylesheet will cause a new `RELS-EXT` Datastream to be created, or will amend the content of the latest revision of the `RELS-EXT` Datastream, if it already exists.

The output will also include a new, generated Service Deployment object (`cmodel-n.deploymentN.xml`) for each one listed in the input's `cmodel-n.deployments.txt` files. This SDep will match the content of the source SDep in the repository, but it will have a different PID, will be in FOXML 1.1 format, and will use input part names consistent with the Datastream IDs specified in the associated content model's `DS-COMPOSITE-MODEL` Datastream.

## 6. Run the Transformer

The transformer applies stylesheets to FOXML stored in a Fedora repository. Although it directly accepts the output of the analyzer and generator, it can also be used outside of the migration process for making low-level changes to batches of objects. When running the transformer in this way, the repository should be shut down, and the `rebuilder` should be run immediately afterward (see section below).

Here's how the transformer works: It scans a directory for PID list files, ending with `.txt`. For each, if a `.xslt` file exists with the same name, that stylesheet is applied to the repository's FOXML for each object in the PID list.

Warning



The transformer makes changes to object XML in the repository. It is possible to damage some, or all of the objects in the repository by using the transformer, so be sure to make a backup first!

### A. Configuring the Transformer

Like the analyzer and generator, the transformer accepts a Java properties file for configuration. The transformer is configured with the same properties that the generator is configured with (see above), and also takes the following:

```
# Whether to run the transformer in "dry run" mode or not.
# In dry run mode, transformation will be tested but no changes will be written
dryRun=true
```

Although not required, it is strongly recommended that you run the transformer with `dryRun=true` the first time, to ensure all transformations will fully succeed.

## B. Transformer Usage

The transformer utility also takes the configuration file as a parameter. For example:


```
java -jar transformer.jar migration.properties
```

Transformation will take roughly double the time that analysis took, since it must read and write each file in the repository.

## 7. Run the Rebuilder

Now that all of your existing objects have been upgraded, you need to run the `rebuilder` so Fedora's database is up-to-date with the files on disk. See the [Rebuilder](#) documentation for further details.

Information

 You MUST rebuild the SQL database. Rebuilding the Resource Index is required only if your Fedora 3.x Repository has been configured to enable the Resource Index.

Rebuilding the SQL database will take around double the time that the transformer took. Rebuilding the Resource Index may take significantly longer.


## 8. Ingest Generated Objects

After the rebuilder has run successfully, you should restart your Fedora 3.x instance and visit the describe page (e.g. <http://localhost:8080/fedora/describe>) again to make sure it started successfully.

Now you'll need to ingest all the new Content Model and Service Deployment objects created during the analyzer and generator steps. To do this:

- Run `fedora-admin.bat` or `fedora-admin.sh` and login to your new repository from the local machine.
- Go to File -> Ingest > Objects from Directory, and choose directory where these files were written (e.g. `C:\fedora-migration`)
- Choose FOXML as the format, and select *OK*.

Information

 The admin client may try to ingest all files in the directory, including the non-FOXML files. While harmless, you may wish to copy all FOXML files (all files in that directory that end in `.xml`) to a separate directory, and ingest from there.

## 10. Verify Success and Clean Up

You should now verify that the upgraded objects are behaving as expected. Pick a few from each `cmodel-n.members.txt` file and do the following for each:

- Visit the object's "Object Profile" page (e.g. <http://localhost:8080/fedora/get/demo:MyPID>)
- From that page, navigate to "View Item Index", and click each Datastream, ensuring it downloads properly.
- If you had Disseminators in your original repository, you should also navigate to the "View Dissemination Index" page and make sure your old Disseminations appear, and that they execute properly.

If you had Disseminators in your original repository, you are now making use of a NEW set of Service Deployment objects. You may now want to purge the original BMEchs, since they are no longer in use. The simplest way to do this is with the `fedora-admin` GUI. The PIDs of the old former BMEchs are enumerated in the file `sdeps.txt`.