Introduction

This is a short manual on how to use the Central Configuration plugin for IBM Content Navigator & IBM Case Manager.

For updates please see:

The introductory blogpost at ivojonker.nl

Also available in .docx and .pdf

Contents

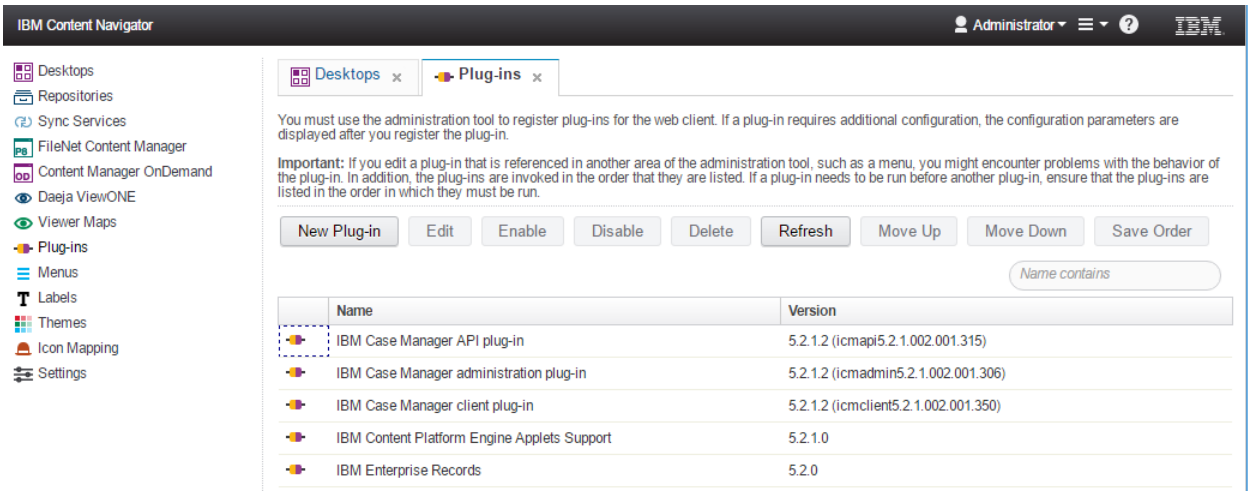
1. Preparing and installing the plugin

1.1. Preparing the .jar file for use in your environment.

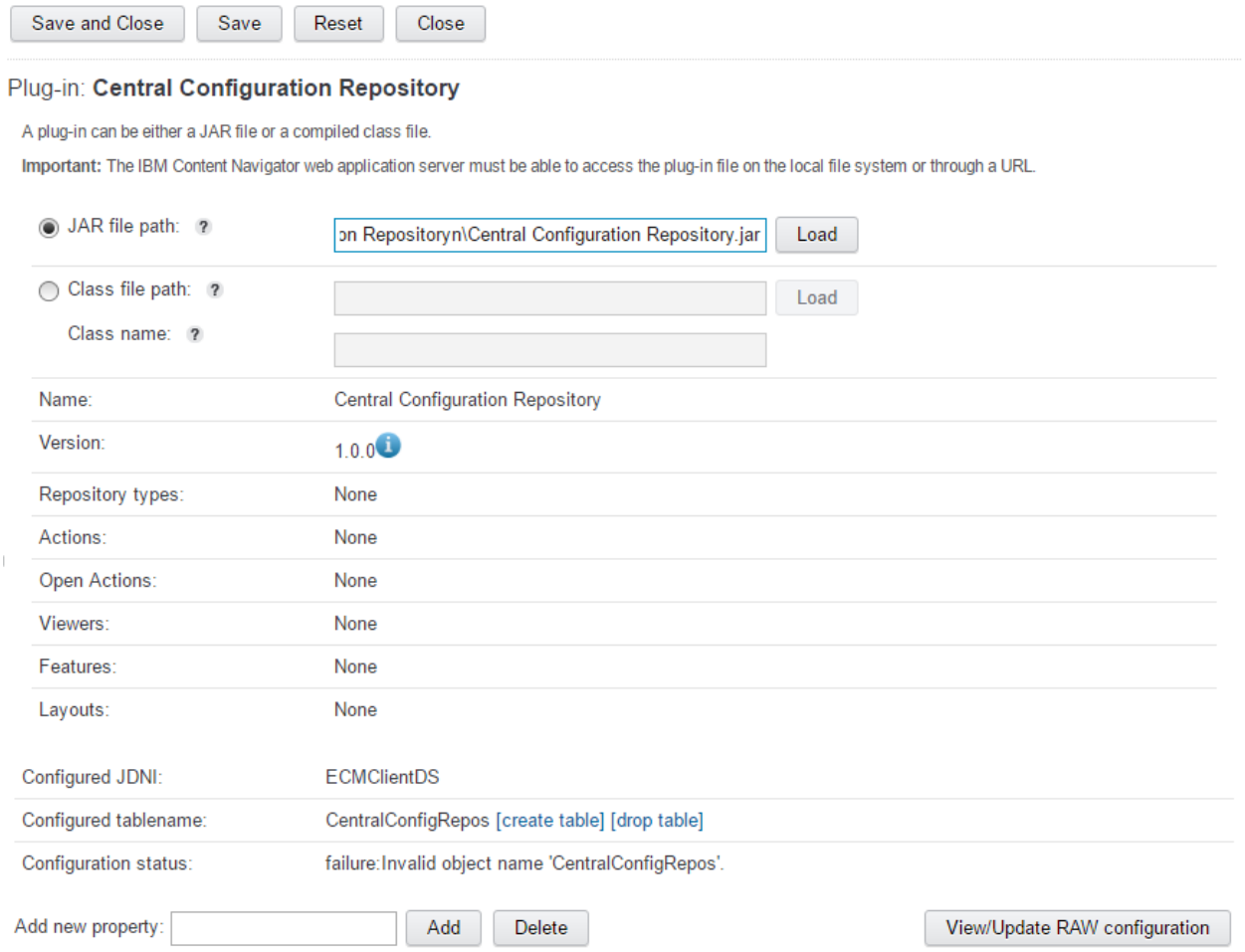
1. Download the CentralConfiguration.jar plugin file.
2. Open the CentralConfiguration.jar plugin file with an archive browser (e.g. WinRAR, or 7zip)
3. Within the CentralConfiguration.jar open the 'config.properties' file for editing.
   Three properties need to be specified, although the defaults may probably work, please take into account the following:
   a. **DATASOURCEJNDI**: The configuration plugin relies on a datasource to be available within the JNDI. While you may re-use the navigator's datasource (default ECMClientDS), it is probably better to define a new – specific datasource for architectural reasons.
   b. **CONFIGTABLENAME**: A database table will be created within the datasource. Be sure not to pick an existing table-name.
   c. **CREATETABLEQUERY**: When creating the table to store the configuration, this query will be used. Update the data-types as needed by your database provider.
4. Save the updated 'config.properties' file within the 'CentralConfiguration.jar'.

1.2. Installing the .jar file as navigator plugin / for use within navigator

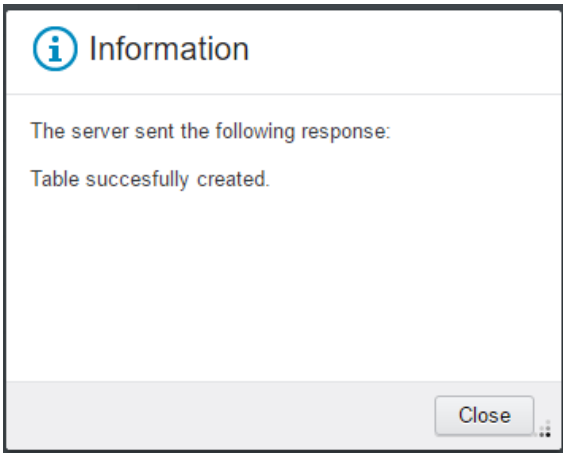
1. Login as an administrative user on the admin desktop (navigator?desktop=admin). Then navigate to the 'Plug-ins' page and press the 'New Plugin-button'.



2. Fill in the path to the JAR file and press load;



3. After loading, look at the field “Configuration status”, if all was configured well there’s probably a notice that the configuration table does not yet exist. Click on the [create table] button. The following message should appear:



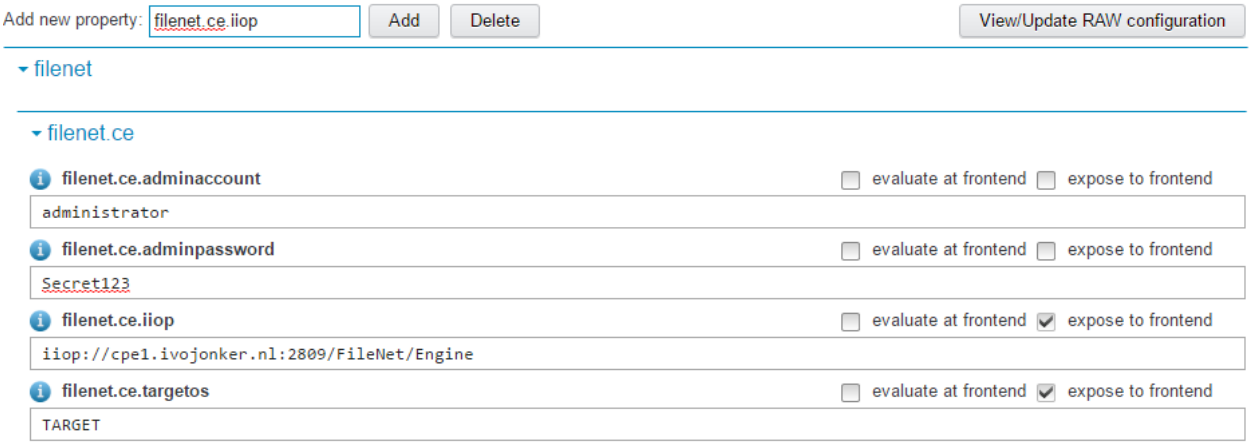
And the status becomes ‘ready’

Configured JDNI:	ECMClientDS
Configured tablename:	CentralConfigRepos [create table] [drop table]
Configuration status:	ready

4. All is set.

1.3. Adding and removing configuration entries using the ICN Plugin manager

1. To add an entry, simply type in your key in the ‘Add new property’ textfield, and press add.
2. The new entry will be automatically grouped with other entries sharing the same namespace.



Each entry has the following attributes:
Key: The identification attribute that will be used to retrieve the property.
Value: The value of the attribute which can be any (multiline) string.
Description: (click on the information icon) A description of the property so other administrators can understand it’s purpose without consulting documentation.
Expose to frontend: Whether or not this entry should be accessible on the frontend. **Warning**: please note that any properties exposed to the frontend will be accessible to all authenticated users. Its discouraged to expose any passwords or api-keys.
Evaluate at frontend: Whether or not to call javascript-eval() on the content when accessing the content via JavaScript. This can be useful if you want to have JSON objects desterilized, or functions as configuration objects.

3. Finally press the “save” button to persist the changes. If all goes well, no messages will be shown.

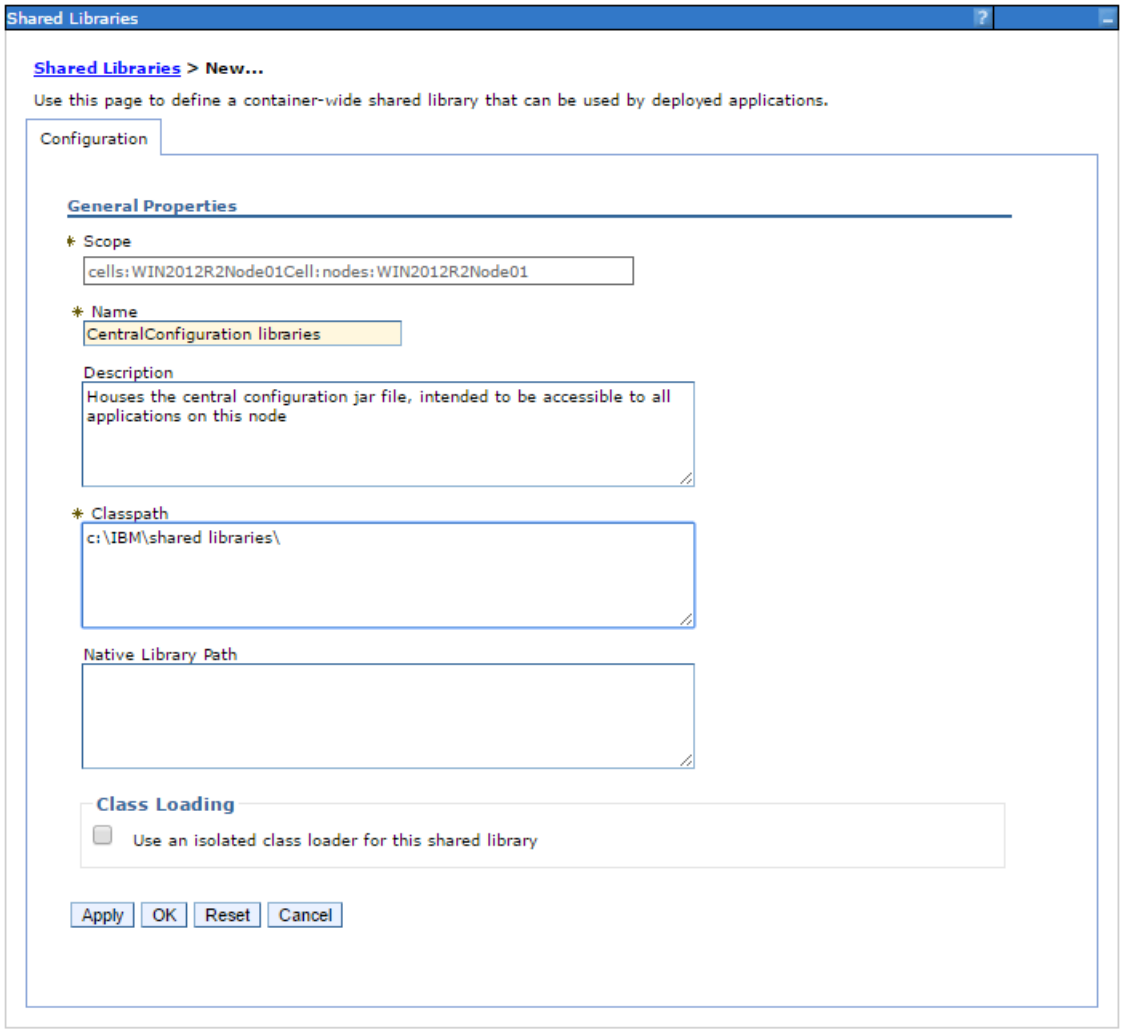
An entry can be removed by entering its full canonical name in the ‘Add new property field’ followed by pressing the ‘Delete’ button.

1.4. Installing the .jar for global use within java EE applications (WebSphere)

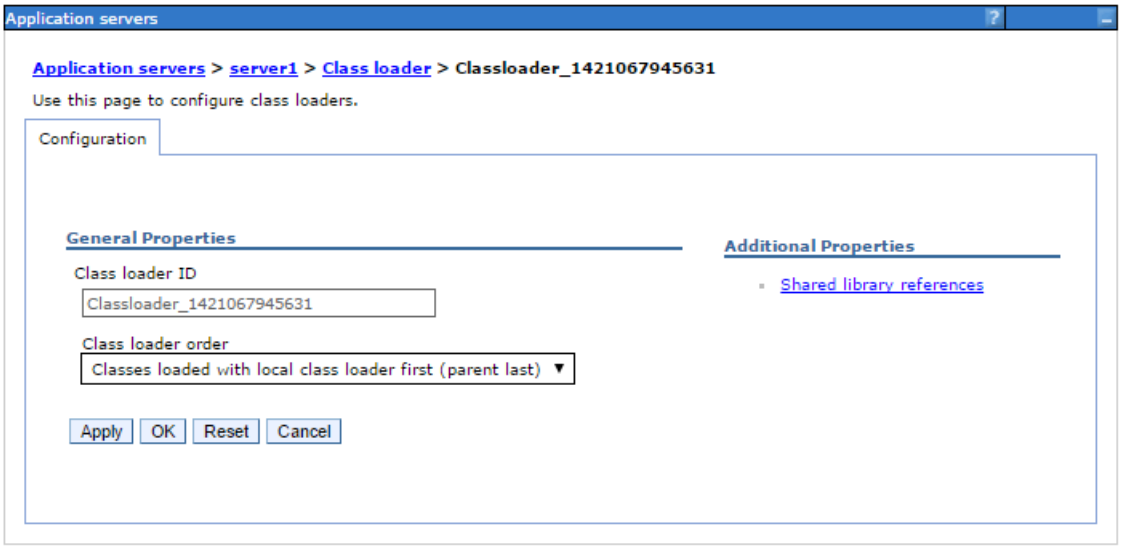
I personally prefer to make the configuration plugin accessible to all the applications running on WebSphere. Only one instance will exist per JVM and only one .jar has to be maintained. The way I usually achieve this is my creating a shared library and connect it to JVM's class loader.

After taking these steps, don't forget to reference a copy of the configuration.jar during design/build-time.

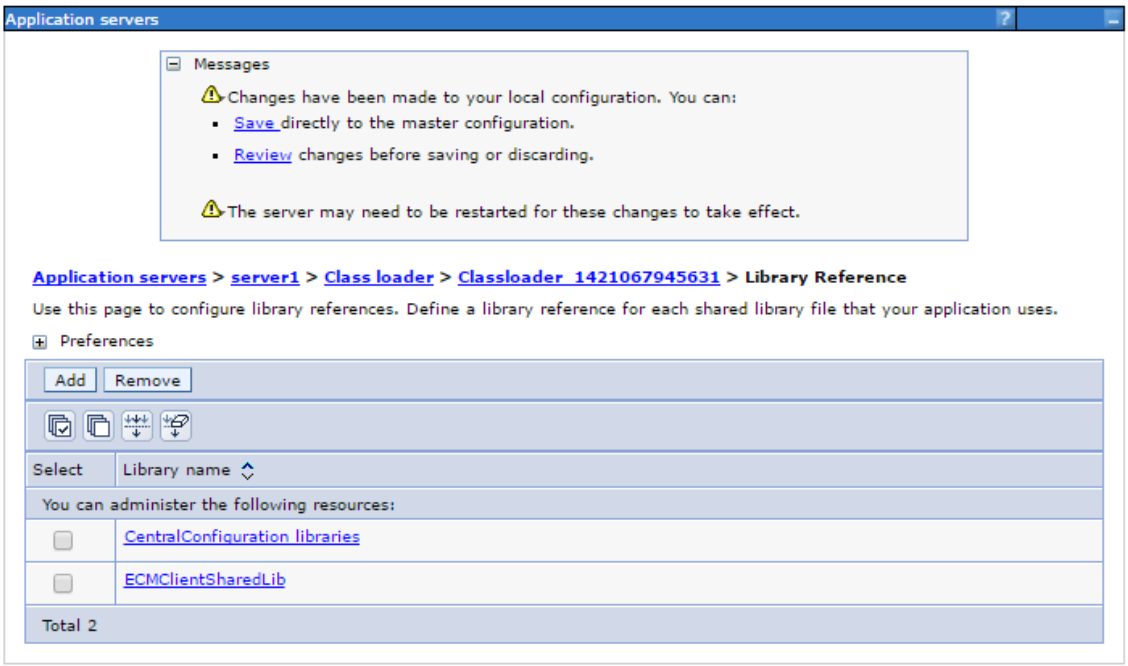
1. Copy the plugin .jar to newly made directory accessible to the WebSphere server. For example, 'c:\IBM\shared libraries\'.
2. Open the WebSphere console (https://localhost:9043/ibm/console) and navigate to Environment -> Shared Libraries. Press "New" to create a new Shared Library.
3. Choose a meaningful name (e.g. "Central Configuration", or "<Project name>Shared libraries" if you're intended on having multiple jars available for general use). Enter your path (from step 1) in the class path.



4. Press apply, followed by Save.
5. Next navigate to Servers-> WebSphere application servers-><server name>->Java and Process Management->Class loader.
6. Select the class loader already available (if no available, create one). Navigate to “Shared library references”.



7. Add the newly created shared library, then press save.



8. Finally restart WebSphere to effectuate.

1.5. Using the .jar in a single java EE application (running on WebSphere)

Simply copy the jar file within your <app.war>\web-inf\lib directory; or your <ear>\lib directory and you’re set to go.

2. Accessing the plugin from your code.

2.1. From dojo/JavaScript.

Assuming you're trying to access specific properties from within the IBM Content Navigator context, the following snippet is all you need to get started.

```
require("CentralConfiguration").get("filenet.ce.iiop")
```

A second parameter can be provided to return a default value if the key ends up to be null or not present in the configuration repository.

2.2. From java

Please take into account that the Central Configuration plugin requires the application to be run within a WebSphere context.

```
package nl.ivojonker.sample;

import nl.ivojonker.icn.configuration.Configuration;

public class SampleUsage {

	public static void doStuff(){

		boolean useCache = false;

		//Retrieves the singleton with default useCache = true Simply
re-uses earlier collected configuration
		Configuration config = Configuration.getInstance();

		//Retrieves the singleton with the option to provide
useCache=false
		config = Configuration.getInstance(useCache);

		//Retrieves the singleton, but will not throw exceptions upon
failure - might sometimes be convenient.
		//The risk of not having a configuration available might be
mitigated by using the following .getProperty(key,default) function.
		config = Configuration.getInstanceSupressExceptions(useCache)

		//retrieves the value for filenet.ce.defaultobjectstore.
		config.getProperty("filenet.ce.defaultobjectstore");


		//retrieves the value for filenet.ce.defaultobjectstore, but
returns default if the value does not exist.
		config.getProperty("filenet.ce.defaultobjectstore","TARGET");
	}

}
```