Apache Karaf Cave 3.x - Documentation

Apache Software Foundation

Apache Karaf Cave 3.x - Documentation

Overview

User Guide

Overview

Apache Karaf Cave is an Apache Karaf sub-project.

It provides an OSGi Bundle Repository (OBR) and Karaf Features Repository (KFR).

OBR provides a service that can automatically install a bundle, with its deployment dependencies, from a bundle repository.

Apache Karaf Cave provides the following features:

- **Storage**: Karaf Cave includes a storage backend. The default one is a simple filesystem backend. As the Cave backend is designed in a plugin way, you can implement your own backend (for instance, JDBC or LDAP backend).
- **OBR Metadata Generation**: Karaf Cave automatically creates the OBR metadata for you, using the artifacts presents in the Cave repository storage.
- **OBR Registration**: Karaf Cave allows you to directly register a Cave repository into an OBR RepositoryAdmin OSGi service.
- **Artifact Upload**: Users can upload OSGi bundle in a Cave repository. It supports URLs like `mvn:groupId/artifactId/version`, `file:`, `http:`, etc.
- **Repository proxy**: Karaf Cave is able to proxy an existing repository, for instance an existing Maven repository. The artifacts are located on the "external" repository, Cave handles the OBR metadata. Cave supports file: and http: URLs, it means that Cave is able to browse a remote HTTP Maven repository for instance.
- **Repository population**: Karaf Cave is able to get artifacts present on an "external" repository (local file: or remote http:), looking for OSGi bundles, and copy the artifacts in the Cave repository storage.

The `cave-server` feature provides the OBR and KFR server, including the storage backend, management layer and shell commands.

User Guide

1. Installation

This chapter describes how to install Apache Karaf Cave into an existing Apache Karaf instance.

1.1. Pre-installation requirements

As Apache Karaf Cave is a Apache Karaf sub-project, it has to be installed into a running Apache Karaf instance.

Apache Karaf Cave is available as Apache Karaf features. The easiest way to install is just to have an internet connection from the Apache Karaf running instance.

1.2. Registration of the Apache Karaf Cave features

Simply register the Apache Karaf Cave features URL in your Apache Karaf instance:

```
karaf@root()> feature:repo-add mvn:org.apache.karaf.cave/apache-karaf-cave/3.0.0/xml/
features
```

Now Apache Karaf Cave features are available, ready to be installed:

```
karaf@root()> feature:list |grep -i cave
cave-server                   | 3.0.0   | x         | karaf-cave-3.0.0 |
```

1.3. Starting Apache Karaf Cave Server

The Apache Karaf Cave Server is installed by the `cave-server` feature:

```
karaf@root()> feature:install cave-server
```

NB: installation of the cave-server feature will install additional features, such as obr, http, war. It could take several minutes depending of your network connection speed.

New Apache Karaf Cave commands are now available:

```
karaf@root()> cave:<TAB>
cave:repositories            cave:repository-create       cave:repository-destroy
cave:repository-install      cave:repository-populate     cave:repository-proxy
cave:repository-uninstall    cave:repository-update       cave:repository-upload
```

2. Repository

A Cave Repository is a container for:

- OSGi bundles (jar files)
- OBR (OSGi Bundle Repository) metadata (aka a repository.xml file)

By default, a repository uses a filesystem backend for the storage, the directory used is KARAF_BASE/cave.

You can change the storage location in the `etc/org.apache.karaf.cave.server.storage.cfg` configuration file:

```
################################################################################
#
#    Licensed to the Apache Software Foundation (ASF) under one or more
#    contributor license agreements.  See the NOTICE file distributed with
#    this work for additional information regarding copyright ownership.
#    The ASF licenses this file to You under the Apache License, Version 2.0
#    (the "License"); you may not use this file except in compliance with
#    the License.  You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing, software
#    distributed under the License is distributed on an "AS IS" BASIS,
#    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#    See the License for the specific language governing permissions and
#    limitations under the License.
#
################################################################################

#
# Storage location where Apache Karaf Cave create repositories by default
#
storage.location=cave
```

For instance, you can define `/var/cave/store` for the `storage.location` property.

2.1. Create

The `cave:repository-create` command creates a new repository:

```
karaf@root()> cave:repository-create my-repository
```

A repository is identified by a name, `my-repository` in our example.

Apache Karaf Cave creates the repository storage for you.

If you want to use an existing directory, and avoid Cave to create one in the storage location, you can use the `-l` (`--location`) option:

```
karaf@root()> cave:repository-create -l /home/user/.m2/repository m2
```

By default, Apache Karaf Cave scans the repository storage and create the OBR metadata. You can use the `-no` (`--no-obr-generate`) option to avoid this step:

```
karaf@root()> cave:repository-create -no -l /home/user/.m2/repository m2
```

By default, Apache Karaf Cave registers (installs) a new repository into the OBR service. You can use the `-ni` (`--no-install`) option to avoid this step:

```
karaf@root()> cave:repository-create -ni -l /home/user/m2/repository m2
```

NB: the `-no` and `-ni` options are interesting when you use an existing location for the repository. If you create a new empty repository, these options don't have really any effect.

2.2. List

You can list the repositories using the `cave:repositories` command:

```
karaf@root()> cave:repositories
Name          | Location
-------------------------------------------------------------------
my-repository | /home/jbonofre/apache-karaf-3.0.1/cave/my-repository
m2            | /home/jbonofre/.m2/repository
```

You can find the repository name and location.

2.3. Repository and OBR service

By default, Apache Karaf Cave installs the repository in the Apache Karaf OBR service.

You can see the installed repositories using the `obr:url-list` command:

```
karaf@root()> obr:url-list
Index | OBR URL
-----------------------------------------------------------------------------
0     | file:/home/jbonofre/.m2/repository/repository.xml
1     | file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/repository.xml
```

You can refresh a repository or install a repository in the OBR service using the `cave:repository-install` command:

```
karaf@root()> cave:repository-install my-repository
```

You can uninstall a repository from the Apache Karaf OBR service using the `cave:repository-uninstall` command:

```
karaf@root()> cave:repository-uninstall my-repository
```

You can see with the `obr:url-list` command that the repository is no more known by the OBR service:

```
karaf@root()> obr:url-list
Index | OBR URL
-----------------------------------------------------------
0     | file:/home/jbonofre/.m2/repository/repository.xml
```

You can "re-install" the repository at any time using the `cave:repository-install` command again:

```
karaf@root()> cave:repository-install my-repository
karaf@root()> obr:url-list
Index | OBR URL
-----------------------------------------------------------------------------
0     | file:/home/jbonofre/.m2/repository/repository.xml
1     | file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/repository.xml
```

At any time, you can force the generation and update of the OBR metadata using the `cave:repository-update` command:

```
karaf@root()> cave:repository-update my-repository
```

This command scan the repository storage location, seeking all OSGi bundles, and generate the OBR metadata (aka a repository.xml).

NB: If you add or remove any artifact in the repository storage location by hand (for instance, using cp, rm, etc on Unix), you have to update the repository OBR metadata using the `cave:repository-update` command.

2.4. Destroy

The `cave:repository-destroy` command deletes the repository storage location:

```
karaf@root()> cave:repository-destroy my-repository
```

Be aware that this command completely deletes the repository storage. By extremely careful when using this command with repository using "custom" location.

2.5. What's next

Apache Karaf Cave provides a HTTP service to expose the repositories. It allows you to use the repositories remotely. You can find details in the HTTP wrapper service section of the user guide.

If repositories are installed in the OBR service, you can use the OBR command as explained in the OBR commands section of the user guide.

3. Populate repository

You can add new artifacts in a repository.

3.1. Upload a single artifact

You can upload a single artifact into a Cave Repository:

```
karaf@root()> cave:repository-upload-artifact my-repository file:/home/jbonofre/.m2/
repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.asm/3.3_2/
org.apache.servicemix.bundles.asm-3.3_2.jar
karaf@root()> cave:repository-upload-artifact my-repository http://svn.apache.org/repos/
asf/servicemix/m2-repo/org/apache/qpid/qpid-broker/0.8.0/qpid-broker-0.8.0.jar
```

You can also use Maven style URL:

```
karaf@root()> cave:repository-upload-artifact my-repository
mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.ant/1.7.0_5
```

3.2. Populate from an external repository

You can also make a kind of "bulk" population of your repository, using an external repository:

```
karaf@root()> cave:repository-populate my-repository file:/home/jbonofre/.m2/repository
```

Apache Karaf Cave supports `file:` but also `http:` URL. It means that Apache Karaf Cave is able to browse a remote repository and copy the artifacts in your "local" repository.

For instance, you can populate your repository using all Ant ServiceMix bundles present on the Central Maven repository:

```
karaf@root()> cave:repository-populate my-repository http://repo1.maven.org/maven2/org/
apache/servicemix/bundles/org.apache.servicemix.bundles.ant/
```

You can also populate with the whole Central Maven Repository:

```
karaf@root()> cave:repository-populate my-repository http://repo1.maven.org/maven2
```

Maven Central repository is really huge and populating from the whole Maven Central Repository will take very very long time. It's just for demonstration purpose.

You can filter the artifacts that you want to pick up to populate the repository. The `cave:repository-populate` command accepts a regex option for the filter. For instance, to pick up only joda-time version 2 artifact, you can run:

```
karaf@root()> cave:repository-populate --filter .*joda-time-2.* my-repository
http://repo2.maven.org/maven2/joda-time/joda-time
```

4. Proxy repository

As you can populate repository, you can also proxy an "external" repository.

It means that the artifacts stay on the remote repository, Apache Karaf Cave generates the OBR metadata in the local repository for the remote artifacts:

```
karaf@root()> cave:repository-proxy my-repository http://repo1.maven.org/maven2/org/
apache/servicemix/bundles/org.apache.servicemix.bundles.commons-lang/
```

NB: the Cave repository will only handle the OBR metadata, it doesn't monitor the remote repository. It means that you have to call the `cave:proxy-repository` command each time the remote repository change (new artifacts, etc).

NB: a best practice is to create a Cave repository dedicated for each proxied repository.

The `cave:proxy-repository` command accepts the filter option, as the `cave:populate-repository` command:

```
karaf@root()> cave:repository-proxy --filter .*joda-time-2.* my-repository
http://repo2.maven.org/maven2/joda-time/joda-time
```

5. OBR commands

To install a repository in the OBR service, you have to use:

```
karaf@root()> cave:repository-install cave-repo
```

Now, you can see the repository OBR metadata register in the OBR service:

```
karaf@root()> obr:url-list
Index | OBR URL
------------------------------------------------------------------------------
0     | file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/repository.xml
```

And the OSGi bundles present in the repository OSGi bundles are available in the OBR service:

```
karaf@root()> obr:list
Name                                          | Symbolic Name
| Version
-------------------------------------------------------------------------------------------------------------
Apache ServiceMix :: Bundles :: commons-dbcp | org.apache.servicemix.bundles.commons-dbcp
| 1.4.0.3
...
```

You can get some detail about an OSGi bundle:

```
karaf@root()> obr:info org.apache.servicemix.bundles.commons-dbcp,1.4.0.3
-------------------------------------------
Apache ServiceMix :: Bundles :: commons-dbcp
-------------------------------------------
id: org.apache.servicemix.bundles.commons-dbcp/1.4.0.3
description: This OSGi bundle wraps commons-dbcp 1.4 jar file.
documentation: http://www.apache.org/
symbolicname: org.apache.servicemix.bundles.commons-dbcp
presentationname: Apache ServiceMix :: Bundles :: commons-dbcp
license: http://www.apache.org/licenses/LICENSE-2.0.txt
uri: file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/
org.apache.servicemix.bundles.commons-dbcp-1.4.0.3.jar
size: 171252
version: 1.4.0.3
Requires:
   package:(&(package=javax.naming))
   package:(&(package=javax.naming.spi))
   package:(&(package=javax.sql))
   package:(&(package=javax.transaction))
   package:(&(package=javax.transaction.xa))
   package:(&(package=org.apache.commons.pool)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.apache.commons.pool.impl)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.xml.sax))
   package:(&(package=org.xml.sax.helpers))
Capabilities:
   bundle:{manifestversion=2, symbolicname=org.apache.servicemix.bundles.commons-dbcp,
presentationname=Apache ServiceMix :: Bundles :: commons-dbcp, version=1.4.0.3}
   package:{package=org.apache.commons.dbcp.cpdsadapter,
uses:=org.apache.commons.dbcp,javax.naming,javax.sql,org.apache.commons.pool.impl,org.apache.commo
version=1.4.0}
   package:{package=org.apache.commons.dbcp,
uses:=org.apache.commons.pool.impl,org.apache.commons.pool,javax.sql,javax.naming,javax.naming.spi
version=1.4.0}
   package:{package=org.apache.commons.dbcp.managed,
uses:=org.apache.commons.dbcp,javax.sql,org.apache.commons.pool.impl,javax.transaction,org.apache.
version=1.4.0}
   package:{package=org.apache.commons.dbcp.datasources,
uses:=javax.sql,org.apache.commons.pool,javax.naming,org.apache.commons.dbcp,javax.naming.spi,org.
version=1.4.0}
   package:{package=org.apache.commons.jocl, uses:=org.xml.sax.helpers,org.xml.sax,
version=1.4.0}
```

NB: in Apache Karaf, the OBR entry format is symbolicname,version

You have the details of the OSGi bundle, especially the requirements and capabilities of the bundle.

OBR is able to resolve the dependencies between bundles, depending of the requirements and capabilities of each bundle. It's able to manage version, etc.

For instance, we have the following commons-dbcp bundle details:

```
karaf@root> obr:info org.apache.servicemix.bundles.commons-dbcp
-------------------------------------------
Apache ServiceMix :: Bundles :: commons-dbcp
-------------------------------------------
id: org.apache.servicemix.bundles.commons-dbcp/1.4.0.3
description: This OSGi bundle wraps commons-dbcp 1.4 jar file.
documentation: http://www.apache.org/
symbolicname: org.apache.servicemix.bundles.commons-dbcp
presentationname: Apache ServiceMix :: Bundles :: commons-dbcp
license: http://www.apache.org/licenses/LICENSE-2.0.txt
uri: file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/
org.apache.servicemix.bundles.commons-dbcp-1.4.0.3.jar
size: 171252
version: 1.4.0.3
Requires:
   package:(&(package=javax.naming))
   package:(&(package=javax.naming.spi))
   package:(&(package=javax.sql))
   package:(&(package=javax.transaction))
   package:(&(package=javax.transaction.xa))
   package:(&(package=org.apache.commons.pool)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.apache.commons.pool.impl)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.xml.sax))
   package:(&(package=org.xml.sax.helpers))
Capabilities:
   bundle:{manifestversion=2, symbolicname=org.apache.servicemix.bundles.commons-dbcp,
presentationname=Apache ServiceMix :: Bundles :: commons-dbcp, version=1.4.0.3}
   package:{package=org.apache.commons.dbcp.cpdsadapter,
uses:=org.apache.commons.dbcp,javax.naming,javax.sql,org.apache.commons.pool.impl,org.apache.commo
version=1.4.0}
   package:{package=org.apache.commons.dbcp,
uses:=org.apache.commons.pool.impl,org.apache.commons.pool,javax.sql,javax.naming,javax.naming.spi
version=1.4.0}
   package:{package=org.apache.commons.dbcp.managed,
uses:=org.apache.commons.dbcp,javax.sql,org.apache.commons.pool.impl,javax.transaction,org.apache.
version=1.4.0}
   package:{package=org.apache.commons.dbcp.datasources,
uses:=javax.sql,org.apache.commons.pool,javax.naming,org.apache.commons.dbcp,javax.naming.spi,org.
version=1.4.0}
   package:{package=org.apache.commons.jocl, uses:=org.xml.sax.helpers,org.xml.sax,
version=1.4.0}
```

We can see that commons-dbcp requires org.apache.commons.pool package (between version 1.3.0 and 2.0.0).

If we take a look on commons-pool bundle details:

```
karaf@root()> obr:info org.apache.servicemix.bundles.commons-pool
-----------------------------------------
Apache ServiceMix :: Bundles :: commons-pool
-----------------------------------------
id: org.apache.servicemix.bundles.commons-pool/1.5.4.3
description: This OSGi bundle wraps commons-pool 1.5.4 jar file.
documentation: http://www.apache.org/
symbolicname: org.apache.servicemix.bundles.commons-pool
presentationname: Apache ServiceMix :: Bundles :: commons-pool
license: http://www.apache.org/licenses/LICENSE-2.0.txt
uri: file:/home/jbonofre/apache-karaf-3.0.1/cave/my-repository/
org.apache.servicemix.bundles.commons-pool-1.5.4.3.jar
size: 97332
version: 1.5.4.3
Capabilities:
   bundle:{manifestversion=2, symbolicname=org.apache.servicemix.bundles.commons-pool,
presentationname=Apache ServiceMix :: Bundles :: commons-pool, version=1.5.4.3}
   package:{package=org.apache.commons.pool.impl, uses:=org.apache.commons.pool,
version=1.5.4}
   package:{package=org.apache.commons.pool, version=1.5.4}
```

This bundle provides package org.apache.commons.pool capability.

It means that if I deploy the commons-dbcp bundle, the OBR should also install the commons-pool bundle:

```
karaf@root()> obr:deploy org.apache.servicemix.bundles.commons-dbcp
Target resource(s):
-------------------
   Apache ServiceMix :: Bundles :: commons-dbcp (1.4.0.3)

Required resource(s):
---------------------
   Apache ServiceMix :: Bundles :: commons-pool (1.5.4.3)

Deploying...done.
```

Done: the OBR has resolved the commons-dbcp requirements using the commons-pool capabilities and so installed the commons-pool bundle.

NB: in the `obr:deploy` command, if you don't mention the version, it will take the highly version available.

6. HTTP wrapper service

When you install the Apache Karaf Cave Server, it starts a HTTP service wrapper.

It means that all artifacts and OBR metadata presents in local repositories are exposed over HTTP.

6.1. OBR metadata access

Assuming that you have the following repositories:

```
karaf@root()> cave:repositories
Name          | Location
-------------------------------------------------------------------
my-repository | /home/jbonofre/apache-karaf-3.0.1/cave/my-repository
m2            | /home/jbonofre/.m2/repository
```

You can access the OBR metadata using the following URL in your favorite browser:

```
http://localhost:8181/cave/m2-repository.xml
```

and

```
http://localhost:8181/cave/my-repository-repository.xml
```

NB: the port 8181 is the default one of the Apache Karaf HTTP service.

You can see that the URL follows the format:

You can note the OBR metadata Cave URL format:

```
http://[cave_server_hostname]:[http_service_port]/cave/[cave_repository_name]-repository.xml
```

It means that you can register the repositories on remote Apache Karaf instances.

In a remote Apache Karaf instance, you just have to install the obr feature and register the HTTP wrapper repository.xml URL:

```
karaf@other()> feature:install obr
karaf@other()> obr:url-add http://cave_server:8181/cave/cave-repo-repository.xml
karaf@other()> obr:url-add http://cave_server:8181/cave/m2-repository.xml
```

6.2. OSGi bundles access

Apache Karaf Cave HTTP wrapper service also provide the OSGi bundles binaries via HTTP.

For instance, you have register the my-repository repository in the OBR service:

```
karaf@root()> cave:repository-install my-repository
```

So you have the following bundles available in the OBR service:

```
karaf@root()> obr:list
Name
| Symbolic Name                                                                                                  | Version
----------------------------------------------------------------------------------------------------------------------
Apache ServiceMix :: Bundles ::
commons-dbcp                                                                                                   |
org.apache.servicemix.bundles.commons-dbcp                                                                   | 1.4.0.3
...
```

If we take a look on the detail of the commons-dbcp bundle:

```
karaf@root()> obr:info org.apache.servicemix.bundles.commons-dbcp
-----------------------------------------
Apache ServiceMix :: Bundles :: commons-dbcp
-----------------------------------------
id: org.apache.servicemix.bundles.commons-dbcp/1.4.0.3
description: This OSGi bundle wraps commons-dbcp 1.4 jar file.
documentation: http://www.apache.org/
symbolicname: org.apache.servicemix.bundles.commons-dbcp
presentationname: Apache ServiceMix :: Bundles :: commons-dbcp
license: http://www.apache.org/licenses/LICENSE-2.0.txt
uri: file:/home/jbonofre/.m2/repository/org/apache/servicemix/bundles/
org.apache.servicemix.bundles.commons-dbcp/1.4_3/
org.apache.servicemix.bundles.commons-dbcp-1.4_3.jar
size: 171252
version: 1.4.0.3
Requires:
   package:(&(package=javax.naming))
   package:(&(package=javax.naming.spi))
   package:(&(package=javax.sql))
   package:(&(package=javax.transaction))
   package:(&(package=javax.transaction.xa))
   package:(&(package=org.apache.commons.pool)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.apache.commons.pool.impl)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.xml.sax))
   package:(&(package=org.xml.sax.helpers))
Capabilities:
   bundle:{manifestversion=2, symbolicname=org.apache.servicemix.bundles.commons-dbcp,
presentationname=Apache ServiceMix :: Bundles :: commons-dbcp, version=1.4.0.3}
   package:{package=org.apache.commons.dbcp.cpdsadapter,
uses:=org.apache.commons.dbcp,javax.naming,javax.sql,org.apache.commons.pool.impl,org.apache.commo
version=1.4.0}
   package:{package=org.apache.commons.dbcp,
uses:=org.apache.commons.pool.impl,org.apache.commons.pool,javax.sql,javax.naming,javax.naming.spi
version=1.4.0}
   package:{package=org.apache.commons.dbcp.managed,
uses:=org.apache.commons.dbcp,javax.sql,org.apache.commons.pool.impl,javax.transaction,org.apache.
version=1.4.0}
   package:{package=org.apache.commons.dbcp.datasources,
uses:=javax.sql,org.apache.commons.pool,javax.naming,org.apache.commons.dbcp,javax.naming.spi,org.
version=1.4.0}
   package:{package=org.apache.commons.jocl, uses:=org.xml.sax.helpers,org.xml.sax,
version=1.4.0}
```

we can see that the URI is `file:/home/jbonofre/.m2/repository/org/apache/servicemix/bundles/org.apache.servicemix.bundles.commons-dbcp/1.4_3/org.apache.servicemix.bundles.commons-dbcp-1.4_3.jar`.

But the HTTP wrapper service also exposes the bundle on:

```
http://localhost:8181/cave/org.apache.servicemix.bundles.commons-dbcp-1.4.0.3.jar
```

Actually, Apache Karaf Cave handles bundle URI relatively to the repository.

So, it means that, if you register the my-repository repository on a remote Apache Karaf instance:

```
karaf@other()> feature:install obr
karaf@other()> obr:url-add http://cave_server:8181/cave/my-repository-repository.xml
```

you can take a look on the commons-dbcp details:

```
karaf@root()> obr:info org.apache.servicemix.bundles.commons-dbcp
-----------------------------------------
Apache ServiceMix :: Bundles :: commons-dbcp
-----------------------------------------
id: org.apache.servicemix.bundles.commons-dbcp/1.4.0.3
description: This OSGi bundle wraps commons-dbcp 1.4 jar file.
documentation: http://www.apache.org/
symbolicname: org.apache.servicemix.bundles.commons-dbcp
presentationname: Apache ServiceMix :: Bundles :: commons-dbcp
license: http://www.apache.org/licenses/LICENSE-2.0.txt
uri: http://localhost:8181/cave/org.apache.servicemix.bundles.commons-dbcp-1.4.0.3.jar
size: 171252
version: 1.4.0.3
Requires:
   package:(&(package=javax.naming))
   package:(&(package=javax.naming.spi))
   package:(&(package=javax.sql))
   package:(&(package=javax.transaction))
   package:(&(package=javax.transaction.xa))
   package:(&(package=org.apache.commons.pool)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.apache.commons.pool.impl)(version>=1.3.0)(!(version>=2.0.0)))
   package:(&(package=org.xml.sax))
   package:(&(package=org.xml.sax.helpers))
Capabilities:
   bundle:{manifestversion=2, symbolicname=org.apache.servicemix.bundles.commons-dbcp,
presentationname=Apache ServiceMix :: Bundles :: commons-dbcp, version=1.4.0.3}
   package:{package=org.apache.commons.dbcp.cpdsadapter,
uses:=org.apache.commons.dbcp,javax.naming,javax.sql,org.apache.commons.pool.impl,org.apache.commo
version=1.4.0}
   package:{package=org.apache.commons.dbcp,
uses:=org.apache.commons.pool.impl,org.apache.commons.pool,javax.sql,javax.naming,javax.naming.spi
version=1.4.0}
   package:{package=org.apache.commons.dbcp.managed,
uses:=org.apache.commons.dbcp,javax.sql,org.apache.commons.pool.impl,javax.transaction,org.apache.
version=1.4.0}
   package:{package=org.apache.commons.dbcp.datasources,
uses:=javax.sql,org.apache.commons.pool,javax.naming,org.apache.commons.dbcp,javax.naming.spi,org.
version=1.4.0}
   package:{package=org.apache.commons.jocl, uses:=org.xml.sax.helpers,org.xml.sax,
version=1.4.0}
```

we can see that the URI is `http://localhost:8181/cave/org.apache.servicemix.bundles.commons-dbcp-1.4.0.3.jar`.

It means that we can use directly `obr:deploy` command:

```
karaf@root> obr:deploy org.apache.servicemix.bundles.commons-dbcp
Target resource(s):
-------------------
   Apache ServiceMix :: Bundles :: commons-dbcp (1.4.0.3)

Required resource(s):
---------------------
   Apache ServiceMix :: Bundles :: commons-pool (1.5.4.3)

Deploying...done.
```

7. Administration

When you install Apache Karaf Cave server, it provides a new CaveServerMBean.

This MBean use the following object name:

```
org.apache.karaf.cave:type=repository,name=*
```

Thanks to this MBean, using any JMX client (like jconsole for instance), you can do all actions as you can using the `cave:*` commands:

- `void createRepository(String name, String location, boolean generateObr, boolean install) throws Exception;`
- `void destroyRepository(String name) throws Exception;`
- `void installRepository(String name) throws Exception;`
- `void uninstallRepository(String name) throws Exception;`
- `void populateRepository(String name, String url, boolean generateObr, String filter) throws Exception;`
- `void proxyRepository(String name, String url, boolean generateObr, String filter) throws Exception;`
- `void updateRepository(String name) throws Exception;`
- `void uploadArtifact(String repository, String artifactUrl, boolean generateObr) throws Exception;`