Apache Karaf Cave 4.x - Documentation

Apache Software Foundation

Apache Karaf Cave 4.x - Documentation

Overview

User Guide

1. Installation
    - 1.1. Pre-installation requirements
    - 1.2. Registration of the Apache Karaf Cave features
    - 1.3. Starting Apache Karaf Cave Server
2. Repository
    - 2.1. Create
    - 2.2. List
3. Populate repository
    - 3.1. Upload a single artifact
    - 3.2. Populate from an external repository
4. Proxy repository
5. HTTP wrapper service
    - 5.1. Repository metadata access
6. Maven wrapper service
7. Administration
    - 7.1. JMX
    - 7.2. REST

Overview

Apache Karaf Cave is an Apache Karaf sub-project.

Cave is an implementation of the OSGi Repository specification, providing additional features such as a complete Maven repository support, a REST API for management, support of remote repository proxy.

Cave deals with the requirements and capabilities of all artifacts.

Apache Karaf Cave provides the following features:

- **Storage**: 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).
- **Repository Metadata Generation**: Cave creates the repository metadata for you, using the artifacts presents in the repository storage.
- **Maven support**: Cave repositories act as a complete Maven repository, allowing you to use Cave directly with Maven.
- **REST API**: Cave provides a REST API to manipulate the repositories.
- **Artifact Upload**: Users can upload OSGi bundle in a Cave repository. It supports URLs like mvn:groupId/artifactId/version, file:, http:, etc.
- **Repository proxy**: 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 repository metadata. Cave supports file: and http: URLs, it means that Cave is able to browse a remote HTTP Maven repository for instance.
- **Repository population**: 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.

Karaf Cave provides two components:

- the cave-server is the full OSGi repository service, including the storage, the management layer, the REST service layer, etc.
- the cave-client is a local repository service proxy that use a remote Cave server (not yet available).

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.

Apache Karaf Cave 4.0.x is designed to work on Apache Karaf 4.0.x.

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 cave 4.0.0
Adding feature url mvn:org.apache.karaf.cave/apache-karaf-cave/4.0.0/xml/features
```

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

```
karaf@root()> feature:list|grep -i cave
cave-server                     | 4.0.0                        |          | Uninstalled |
karaf-cave-4.0.0 |
cave-storage                    | 4.0.0                        |          | Uninstalled |
karaf-cave-4.0.0 |
cave-http                       | 4.0.0                        |          | Uninstalled |
karaf-cave-4.0.0 |
cave-rest                       | 4.0.0                        |          | Uninstalled |
karaf-cave-4.0.0 |
cave-maven                      | 4.0.0                        |          | Uninstalled |
karaf-cave-4.0.0 |
```

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
```

The cave-server feature is a meta-feature which actually installs:

- cave-storage feature providing the Cave filesystem default storage.

- cave-http feature providing the Cave HTTP service allowing a remote access to the repositories.
- cave-rest feature providing the Cave REST API allowing to manipulate the repository remotely with any REST HTTP client.
- cave-maven feature providing a complete Maven repository for the Cave repositories.

After the installation of the cave-server feature, new commands are 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:

- Artifacts (files)
- Repository metadata

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
#
cave.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 repository metadata. You can use the `-no` (`--no-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 repository service. You can use the `-ns` (`--no-start`) option to avoid this step:

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

```
karaf@root()> cave:repositories
Name          | Location
-----------------------------------------------------------------
my-repository | /opt/apache-karaf-4.0.0/data/cave/my-repository
```

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 my-repository file:/home/user/.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 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 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/user/.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 repository 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 repository 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:repository-proxy` command accepts the filter option, as the `cave:repository-populate` command:

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

5. HTTP wrapper service

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

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

5.1. Repository metadata access

Assuming that you have the following repositories:

```
karaf@root()> cave:repositories
Name          | Location
-----------------------------------------------------------------------------
my-repository | /opt/apache-karaf-4.0.0/data/cave/my-repository
```

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

```
http://localhost:8181/cave/http/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:

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

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

6. Maven wrapper service

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

It means that all artifacts are available using a Maven structure (groupId/artifactId/version/artifactId-version[-classifier].type).

It allows you to use your Cave repository directly using mvn URL in Karaf, and using Maven itself: Cave acts as a complete Maven repository.

For instance, we have the following Cave repository:

```
karaf@root()> cave:repositories
Name          | Location
-----------------------------------------------------------------
my-repository | /opt/apache-karaf-4.0.0/data/cave/my-repository
```

You can access the corresponding Maven repository using:

```
http://localhost:8181/cave/maven
```

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

You can see that the URL follows the format:

```
http://[cave_server_hostname]:[http_service_port]/cave/maven/
```

For instance, if a Cave repository contains the commons-lang 2.6 artifact, it’s accessible using:

```
http://localhost:8181/cave/maven/commons-lang/commons-lang/2.6/commons-lang-2.6.jar
```

7. Administration

7.1. JMX

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

This MBean uses 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 generate, 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 generate, String filter) throws Exception;
- void proxyRepository(String name, String url, boolean generate, String filter) throws Exception;
- void updateRepository(String name) throws Exception;
- void uploadArtifact(String repository, String artifactUrl, boolean generate) throws Exception;

7.2. REST

Cave provides a complete REST API to manipulate the repositories.

The API is available on:

```
http://localhost:8181/cave/rest
```

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

You can get the WADL:

```
http://localhost:8181/cave/rest?_wadl
```

Last updated 2015-08-03 11:25:24 CEST