Infrastructure - Task #2026

Story # 2025 (Closed): Tool to list versions of dataone components installed on CN

Outline design of the version list tool

2011-11-14 20:10 - Dave Vieglais

Status:	Closed	**Start date:**	2011-11-14
Priority:	Normal	**Due date:**	
Assignee:	Skye Roseboom	**% Done:**	100%
Category:	Management	**Estimated time:**	0.00 hour
Target version:			
Milestone:	CCI-1.0.0	**Story Points:**	
Product Version:	*		

Description

Outline the overall design of a tool that can list versions of each component deployed on the CN.

- What mechanism is used to record the version in jars?
- Which jars need to be inspected?
- What needs to be done to libraries, applications to ensure this information is available?
- How to maintain a list of applications / jars that need to be inspected (config file?)
- Where the tool should be deployed ?

History

#1 - 2011-11-15 21:53 - Skye Roseboom

D1 version, SVN revision, source branch, build time information are now present in jar/war manifest files. This seems to be enough information to determine compatibility mismatches.

#2 - 2011-11-16 14:44 - Skye Roseboom

- Status changed from New to In Progress

#3 - 2011-11-16 15:09 - Skye Roseboom

CN Components and common dependencies:

CN web app (d1_cn_rest) - depends on d1_common_java, d1_libclient_java, d1_portal, d1_cn_common, d1_cn_noderegistry, d1_cn_rest_proxy, d1_identity_manager

KNB web app (metacat) - depends on d1_common_java, d1_libclient_java

portal web app (d1_portal_servlet) - depends on d1_common_java, d1_libclient_java, d1_portal, d1_test_resources
d1_portal - depends on d1_common_java, d1_libclient_java

d1_process_daemon - depends on d1_replication, d1_synchronization
d1_replication - depends on d1_cn_noderegistry, d1_cn_common, d1_libclient
d1_synchronization - depends on d1_libclient, d1_cn_noderegistry

Mercury??
d1_cn_index_processor - depends on d1 indexerapi
indexerapi (d1_indexer2)- depends on d1_common, d1_libclient

d1_noderegistry - depends on d1_cn_common
d1_cn_common - depends on d1_common_java

#4 - 2011-11-16 15:29 - Dave Vieglais

Perhaps this can be simplified so that initially at least, the tool runs from the command line and accepts a single .jar file name as a parameter, and lists the version info for the contained classes?

Will java.lang.Package.getImplementationVersion() and getSpecificationVersion() do the job?

Then we could have a bash script which runs the tool for each of the dataone apps that needs to be examined to compile a report?

#5 - 2011-11-16 21:26 - Skye Roseboom

I'll start by creating a simple command line tool that will report on d1 jars used in war applications.

After looking at the maven-shade plugin generated 'uber' jar, it is becoming clear the implementation strategy for war, jar, shade jars will be different - in order to mine the version information from the proper location in each type of archive.

I also noticed today the manifest.mf file generated in the d1_process_daemon did not contain all the same information available d1_common_java, d1_libclient_java etc. So may need to update some mavin pom config as part of this story

d1_process_daemon's manifest.mf file looks to be controlled by the maven-shade plugin.

d1_replication, d1_synchronization currently do not fill in the manifest file with scm build number, source branch, or build time. The version snapshot is present

#6 - 2011-11-17 14:50 - Skye Roseboom

Initially considering whether it is feasible to eventually add this utility to the cn buildout process so it would be installed along with the cn components. This would ensure the tool is always available for cn environments.

Upgrading the reporting tool would likely need to be a separate process.

#7 - 2011-11-22 14:47 - Skye Roseboom

- Status changed from In Progress to Closed