

CRX Content Package Deployer Plugin

for Jenkins

Mark Adamcin
Acquity Group
Part of Accenture Interactive
mark.j.adamcin@accenture.com

About Me

- Technical Consultant with Acquity Group since 2007, which is now part of Accenture Interactive
- CQ5 Developer and Application Architect since 2009
- Now a Release Management specialist with our Shared Services DevOps group, focused on supporting client application deployments using Jenkins
- Since 2009, I've written five (5!) maven plugins just for building and deploying CRX Content Packages.

This plugin, what is it good for?

- Uploading/Installing CRX Content Packages on CQ/AEM servers
- Downloading CRX Content Packages from CQ/ AEM servers
- Doing both of those things without knowing the AEM admin password
- And it's completely free to use!

Wait, no admin password, you say?

- The plugin uses the httpsig-java implementation of a draft HTTP Signature Authentication scheme based on SSH Public Key login.
- Simply install two OSGi bundles and an authorized_keys file on the AEM instance's filesystem to enable support on the server.
- https://github.com/adamcin/ net.adamcin.sling.auth.httpsig/wiki/getting-started

But why not Maven?

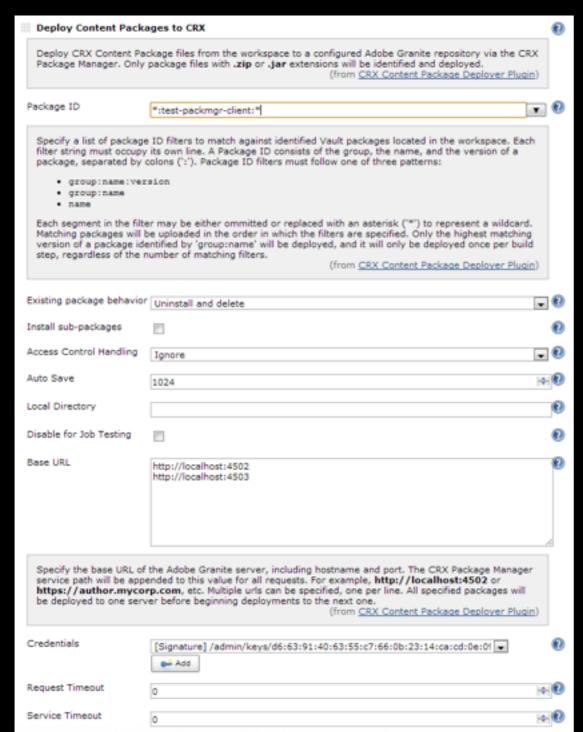
- Maven plugins like the content-package-mavenplugin and maven-vault-plugin generally only support installing one package on one host per execution.
- Managing additional packages and hosts requires a pom.xml file, usually managed separately from Jenkins in SCM.
- Maven logs are full of non-package-deploymentrelated verbosity that gets in the way of knowing whether or not installation truly succeeded.

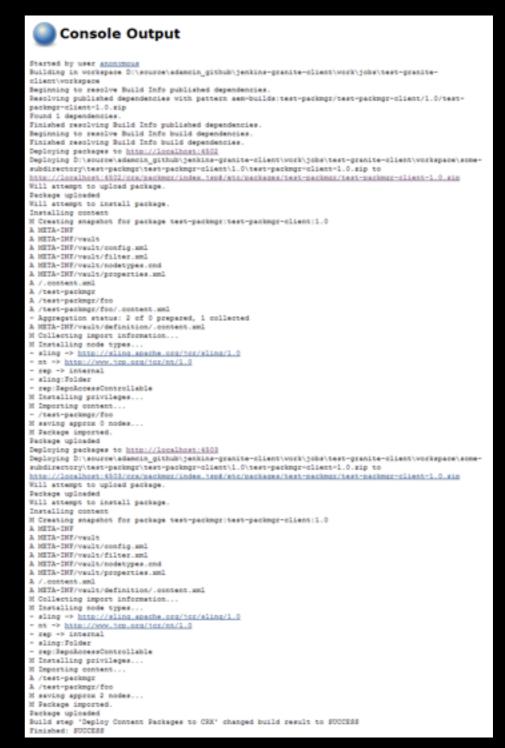
Features

Deploy Packages

- The "Deploy Content Packages to CRX" build step can be added to any job type.
- Flexible in regards to how the package files are created and organized within the workspace, because it will recursively scan and identify any files which have a ".jar" or ".zip" extension.
- Feel free to use this step immediately after a maven build, or after the "Download Content Packages from CRX" step, or after resolving maven artifacts from a repository manager into the workspace.
- Once a package has been identified (i.e. has a valid group:name:version), the
 package will be checked against the Package ID filter to determine whether it
 must be uploaded to the configured Base URL(s).
- If a package installation succeeds with errors "(check logs!)", the build will be marked as unstable.

Deploy Packages

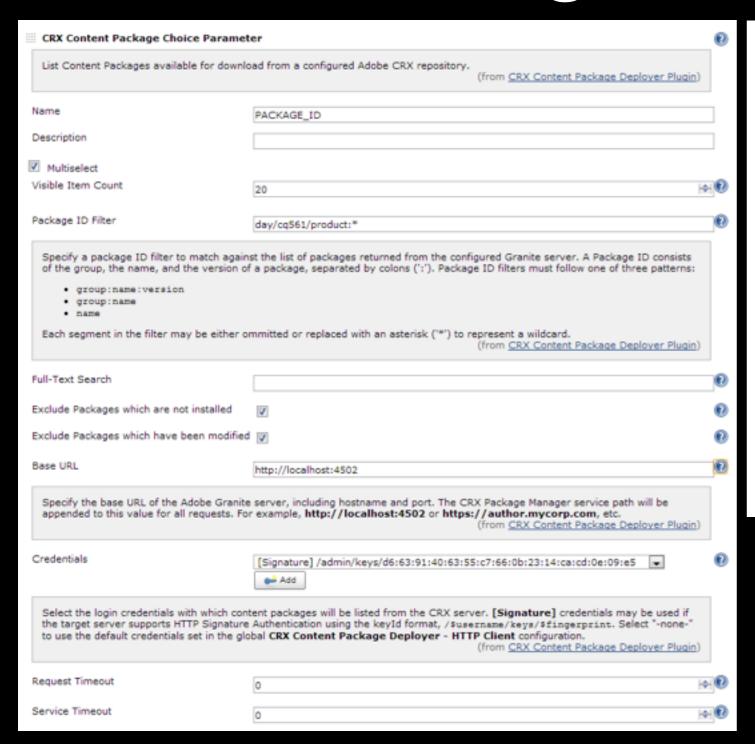


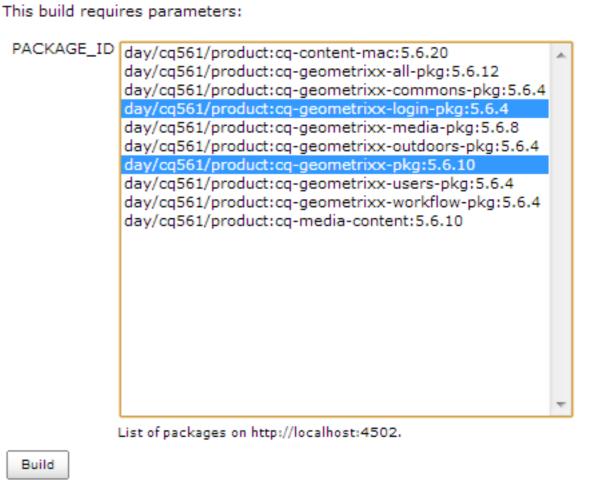


List Packages to Download

Select any number of content packages available for download from a CRX server as a Build Parameter, which can then be used in other components provided by this plugin.

List Packages to Download

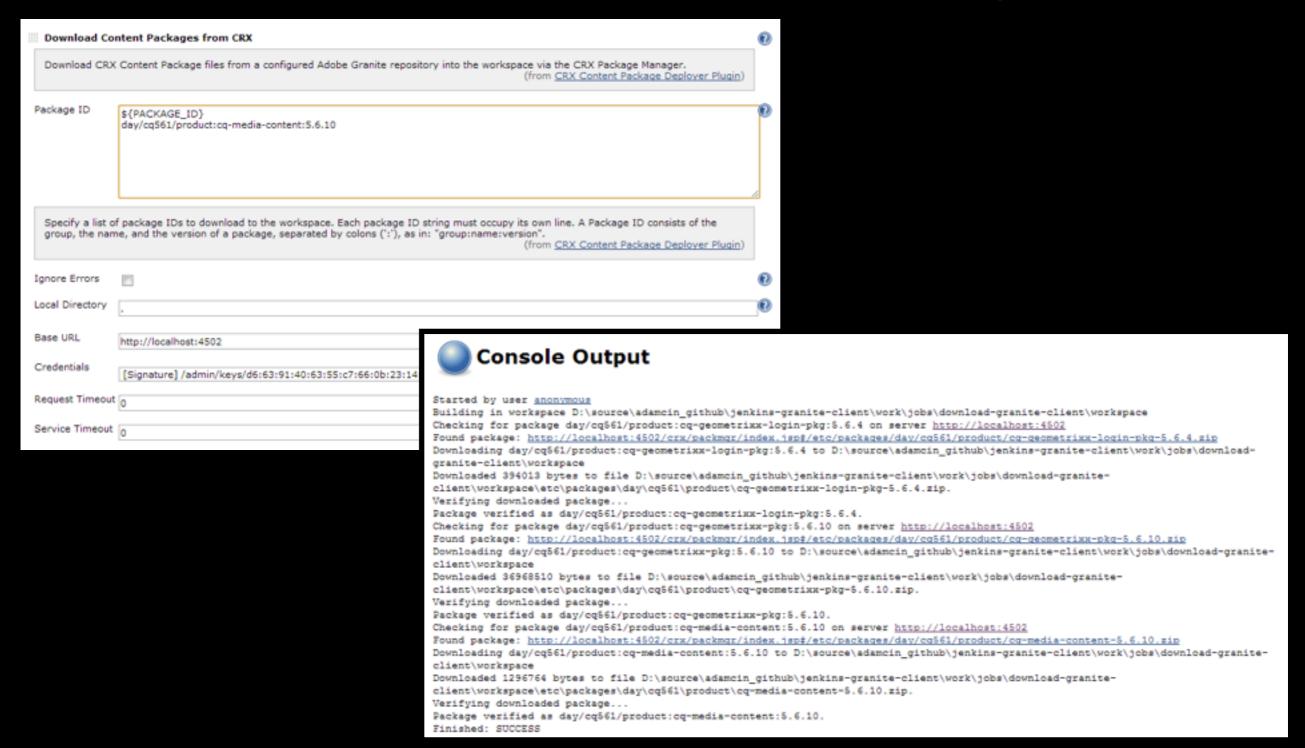




Download Packages

- The "Download Content Packages from CRX" build step can be used on any job type to download content packages to the workspace.
- Use the CRX Content Package Choice Parameter to select one or more packages from a live CRX Package Manager service.
- Downloaded packages are organized according to their CRX installation path, relative to the workspace path, or local directory, if specified.

Download Packages



Compact Workspace Filters

- each line consists of significant text before an optional comment character (#)
- each line that begins with a "/" begins a new filter root.
- the first non-empty, non-comment line must define a new filter root
- each non-empty, non-comment line after a filter root that begins with a + or - defines an include or exclude rule, respectively.
- everything following the + or must be a valid regular expression

Compact Workspace Filters

For example, to include everything under /etc except for packages:

```
/etc # define /etc as the filter root
+/etc(/.*)? # include everything under /etc
-/etc/packages(/.)? # exclude package paths
```

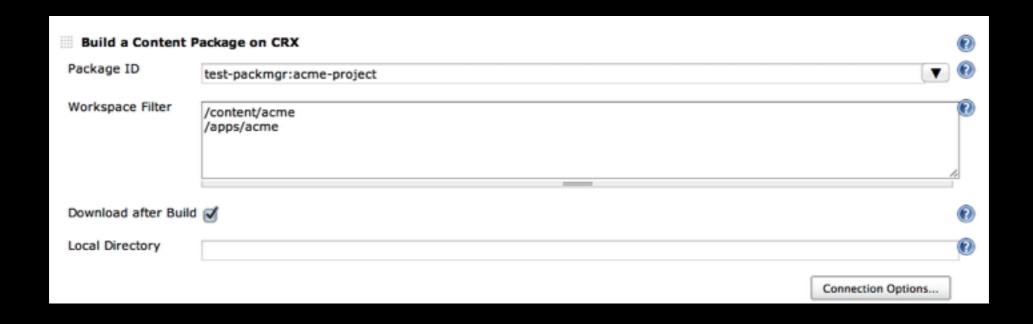
To create a package for a project "acme" defined in CRX DE Lite, a filter may look like this:

```
/content/acme  # include the site content
/apps/acme  # include the app code
```

Build Package

- The "Build a Content Package on CRX" build step can be added to any job type.
- Specify a Package ID (group:name:version) and a Workspace Filter (using the compact syntax) and the package will be created and built on the specified server.
- If built successfully, the package is downloaded to the workspace, from where it can then be archived or uploaded to other servers.
- This step was designed to support an automated refresh of lower environments using production content.

Build Package



Build Package

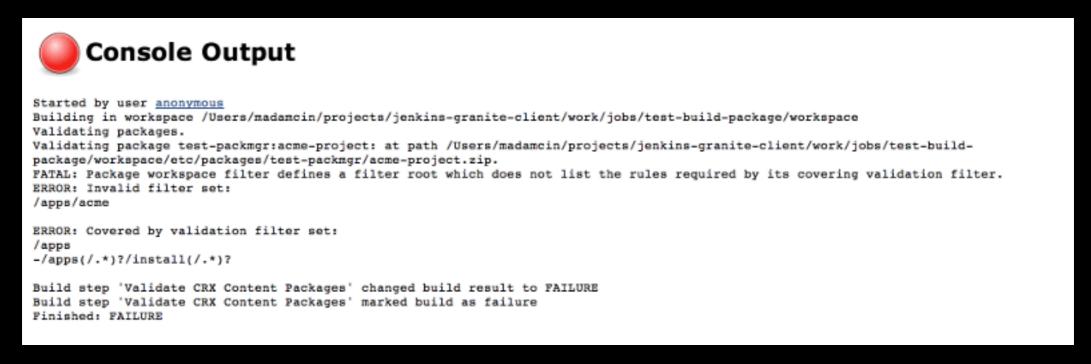
```
Console Output
Started by user anonymous
Building in workspace /Users/madamcin/projects/jenkins-granite-client/work/jobs/test-build-package/workspace
Checking for package test-package:acme-project: on server http://localhost:4502
Found package: http://localhost:4502/crx/packmqr/index.isp#/etc/packages/test-packmqr/acme-project.zip
Package updated successfully
Building package test-packmgr:acme-project:.
Building package
A META-INF
A META-INF/wault
A META-INF/vault/config.xml
A META-INF/vault/filter.xml
A META-INF/vault/nodetypes.cnd
A META-INF/vault/properties.xml
A /.content.xml
A /apps
A /apps/.content.xml
A /apps/acme
A /apps/acme/components
A /apps/acme/components/sample
A /apps/acme/components/sample/html.jsp
A /apps/acme/install
A /apps/acme/install/com.example.acme.jar
A /apps/acme/src
A /apps/acme/src/impl
A /apps/acme/src/impl/com.example.acme.bnd
A /apps/acme/src/impl/libs
A /apps/acme/src/impl/src
A /apps/acme/src/impl/src/main
A /apps/acme/src/impl/src/main/java
A /apps/acme/src/impl/src/main/java/com
A /apps/acme/src/impl/src/main/java/com/example
A /apps/acme/src/impl/src/main/java/con/example/acme
A /apps/acme/src/impl/src/main/java/com/example/acme/SampleUtil.java
A /apps/acme/src/impl/src/main/java/con/example/acme/impl
A /apps/acme/src/impl/src/main/java/con/example/acme/impl/Activator.java
A /apps/acme/src/impl/src/main/resources
A /content
A /content/.content.xml
A /content/acme
A /content/acme/.content.xml
- Aggregation status: 23 of 19 prepared, 21 collected
A META-INF/vault/definition/.content.xml
Package location: http://localhost:4502/crx/packmgr/index.isp#/etc/packages/test-packmgr/acme-project.zip
Downloading test-packmgr:acme-project: to /Users/madamcin/projects/jenkins-granite-client/work/jobs/test-build-package/workspace
Downloaded 11263 bytes to file /Users/madamcin/projects/jenkins-granite-client/work/jobs/test-build-package/workspace/etc/packages/test-
packngr/acme-project.zip.
Verifying downloaded package...
Package verified as test-packmgr:acme-project:.
Finished: SUCCESS
```

Validate Packages

- The "Validate CRX Content Packages" build step can be added to any job type.
- Scan packages for forbidden file extensions and restrict the scope of package workspace filters to control the content that can be deployed along any given pipeline
- Specify a Workspace Filter using the compact syntax, and only the specified filter roots will be allowed. You can even specify mandatory include/exclude rules for each root to ensure that
- This step was designed to support an automated refresh of lower environments using production content.

Validate Packages

Walidate CRX Content P	Packages	•
Package ID	**/acme-*.zip	▼ 0
Local Directory		0
Forbidden Extensions	.zip	Ø
		//
Validation Workspace Filter	/apps -/apps(/.*)?/install(/.)?	0
Allow Non-covered Roots	€	0



Other Features

- Optionally replicate packages after deployment, or use the separate Replicate Content Packages build step
- Use Jenkins Credentials to define Domain Scopes if deploying to multiple environments with different credentials
- Use Jenkins SSH User Private Key Credentials instead of the admin password to avoid password management headaches.
- Use \${TOKENS} in Base URL and Package ID fields to parameterize deployment jobs

Before you deploy your first package, remember...

- Use the "Disable for Job Testing" option to execute a dry run to identify all the packages that would be installed during normal execution
- Clean up your left-over packages from your deploy job workspace when not using SCM or mvn clean!

	Set the project description from a file in the workspace (à la GitHub README.md)	1.1	
✓	Workspace Cleanup Plugin		
	This plugin deletes the workspace before the build or when a build is finished and artifacts saved.	0.19	
	Config File Provider Plugin		

Build Environment

Delete workspace before build starts

Future Features

- Replicate Packages on Install, or as a separate build step (v 1.1)
- Create/Build Packages using a simple text format for specifying a workspace filter (v 1.2)
- Automatic deletion of old package versions

Get It

- Download Jenkins WAR from http://jenkins-ci.org/
- Install the CRX Content Package Deployment Plugin from the Plugin Manager

	A Jenkins CI plugin for uploading dSYM files to Crittercism.	1.1
⋖	CRX Content Package Deployer Plugin Deploys content packages to Adobe CRX applications, like Adobe CQ 5.4, CQ 5.5, and AEM 5.6. Also allows downloading packages from one CRX server and uploading them to one or more other CRX servers.	1.0
	Deploy Plugin This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build	1.9

Install without restart

Download now and install after restart



Read the Wiki!

 https://wiki.jenkins-ci.org/display/JENKINS/ CRX+Content+Package+Deployer+Plugin



Get the Source!

- https://github.com/jenkinsci/crx-contentpackage-deployer-plugin
- https://github.com/adamcin/granite-clientpackman
- https://github.com/adamcin/httpsig-java



Get this Deck!

http://adamcin.net/files/jenkins_crx_plugin.pdf

Mark Adamcin
Acquity Group
Part of Accenture Interactive
mark.j.adamcin@accenture.com