CRX Content Package Deployer Plugin

Plugin Information
View CRX Content Package Deployer on the plugin site for more information.

CRX Content Package Deployer Plugin

Deploys content packages to Adobe CRX applications, like Adobe CQ/AEM. Also allows downloading packages from one CRX server and uploading them to one or more other CRX servers.

Features

- Supports password-less HTTP Signature authentication using your configured SSH Private Keys, eliminating the need to coordinate password exchange between developers, devops, and operations teams.
- Downloads and/or deploys 1-to-many CRX packages per execution, and deploys each CRX package to 1-to-many servers per execution
- The multiselect-enabled Package Choice Parameter allows for execution of parameterized builds using a selection widget populated with a list of packages retrieved from a CRX server.
- Build and download a content package from an AEM instance (for automating production content refresh to lower environments)
- Validate CRX packages prior to an archive or deployment step (check the package workspace filter against a validation workspace filter, and scan the contents of the zip file for forbidden file extensions, like embedded bundles or packages, or even JSP files, if you feel very strongly that your developers should be using EcmaScript exclusively).
- Replicate a content package from CRX (for example, if direct upload to publish servers isn't possible from your Jenkins server)
- Specify workspace filters in the Build and Validate steps using a compact text format that doesn't require escaping regular expressions.

Credentials Support

CRX login credentials are managed using functionality provided by the Credentials Plugin. Users are encouraged to provide a description for each set of credentials and to organize their credentials using Domains, which are regularly filtered by this plugin's components according to Base URL parameters. These practices reduce confusion while increasing reusability and security.

(Since 1.3) If login credentials for a server are different than those configured in the Connection Options section, you may override them in the Base URL by inserting `username[:password]@` between the scheme and the hostname.

For example, to override the Username without changing the associated password or private key, you may use the following form:

```
http://deployer@localhost:4502
```

To override the credentials completely, provide a username and password (which may be provided by an encrypted parameter) by separating them with a colon, as shown below:

```
http://deployer:Password123@localhost:4502
```

The URL in this form is ultimately parsed using the `java.net.URI` class, and therefore special characters should be percent-encoded before use in a Base URL field.

However, because it will be common to accept a password as a job parameter or a global parameter, and many password schemes require the use of at least one special character, not to mention those which allow any character under the sun, the user info part is first sanitized in the follow way before being parsed as a URI:

- Percent characters (`%`) which are unambiguously NOT used to denote escaped ASCII characters (using the regular expression, `%(?![A-Fa-f0-9]{2})`) are replaced with `%25`.
- Reserved URL characters in the set `" !#$\'()*+,/:;=?@[]"` are replaced by a `%` and followed by the correct hexadecimal ASCII code.

This sanitization procedure will result in correct percent-encoding of most reserved characters, and it will not over-escape input which has already been properly percent-encoded. There are a few exceptions, however:

1. The username MUST NOT contain an unescaped colon (`:`).
2. The username and the password MUST NOT contain an unescaped forward-slash (`/`).

3. Neither the username nor the password should contain an unescaped sequence of the percent character (%) followed by two characters in the space of hexadecimal digits (`[A-Fa-f0-9]{2}`), because this sequence will be interpreted as an escaped ASCII character and left unchanged prior to parsing by `java.net.URI`.

Once parsed, the user info part will be stripped from the base URL before it is used by this plugin's components, to prevent credentials from being leaked in the console log. However, care should always be taken when passing credentials through Jenkins parameters in case they are exposed in other areas of the application. In addition, the Base URL field is persisted as plaintext on disk, so any unencrypted passwords stored in that field are visible to anyone who has access to the Jenkins filesystem. Use the Password Parameter type and the Mask Passwords Plugin whenever possible to properly secure your parameterized CRX application credentials.

HTTP Signature Support

In addition to username/password login, this plugin also supports HTTP Signature client authentication using SSH User Private Key credentials, which in turn is provided by SSH Credentials Plugin. The server side of this functionality can be setup in Adobe Granite/CRX servers by installing the Sling HTTP Signature Authentication Handler.

See also:

- Joyent's Specification for HTTP Signature Authentication
- HTTP Signature Authentication for Java

Compact Workspace Filters

Specify a package workspace filter in a simple text format that will be applied to the package before build. This defines what content will be captured by the package.

Simple spec is defined simply as a line-by-line format where:

- 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

Content is "covered" by a filter root if the content path starts with, or is the same as the filter root path. Once the "covering" filter root is identified, its include/exclude rules are evaluated from top to bottom, by matching the content path against each rule pattern (which is a Java regular expression). The last rule whose pattern matches the content path defines whether that content is included (line starts with +) or excluded (line starts with -). If no rules are defined, all content covered by that root path is included

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
```

Components

- #Deploy Content Packages to CRX
- #CRX Content Package Choice Parameter
- #Download Content Packages from CRX
- #Replicate Content Packages from CRX
- #Build a Content Package on CRX
- #Validate CRX Content Packages

Deploy Content Packages to CRX

Summary

The "Deploy Content Packages to CRX" build step can be added to any job type to upload and install CRX content packages from the workspace, or a local directory, if specified. The logic is designed to be flexible in regards to how the package files are created and

organized within the directory, 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.

Configuration

Console Output

CRX Content Package Choice Parameter

Summary

The CRX Content Package Choice Parameter is a build parameter which allows a user to select any number of content packages available for download from a CRX server as as a valid, newline-separated Package Id field value, which can then be used in other components provided by this plugin.

Configuration

Build with Parameters

Download Content Packages from CRX

Summary

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.

Configuration

Console Output

Replicate Content Packages from CRX

Summary

The "Replicate Content Packages from CRX" build step can be used on any job type to asynchronously replicate content packages from one or more Granite servers to their configured replication targets, such as AEM publish servers. Use the CRX Content Package Choice Parameter to select one or more packages from a live CRX Package Manager service. This is probably not the most deterministic way to install packages on publish servers, so prefer using the "Deploy Content Packages to CRX" step whenever possible.

Configuration

Console Output

Build a Content Package on CRX

Summary

Create and build a content package on CRX by specifying a workspace filter, and then download it to the workspace to get archived as a backup, or as the beginning of a magical content package adventure, full of drama, suspense, and deployments to lower environments!

Configuration

Console Output

Validate CRX Content Packages

Summary

Validate that content packages in the workspace conform to restrictions on AC Handling Mode, Filter Root Path Prefixes, Path Inclusion, as well as to restrictions on the scope of their WorkspaceFilter and on the types of embedded files. Use this to enforce security policies to prevent developers from deploying content or code through continuous integration. Only package files with **.zip** or **.jar** extensions will be identified and validated.

Configuration

Console Output

Version History

Version 1.8.1 (Jan 12, 2017)

- Merged PR#8 support for JEP-200 serialization/XStream whitelist change in Jenkins core.

Version 1.8 (Dec 20, 2017)

- Added support for Pipeline and Jenkinsfile. Use the Pipeline DSL Snippet Generator to generate step syntax using the new `crxBuild`, `crxDeploy`, `crxDownload`, `crxReplicate`, and `crxValidate` symbols.
- Updated and explicitly specified dependencies to eliminate outdated versions of async-http-client from being deployed when newer, bug-fixed versions are desired.

Version 1.7.2 (Aug 4, 2017)

- Merged PR#7 "fixed setting custom timeout value (JENKINS-29719)".
- Refactored GraniteClientExecutor to eliminate attempt to access to non-serializable state from within executable body when Credentials are not supplied.
- Moved to ci.jenkins.io and resolved error with package path filter when executing jobs on Windows.

Version 1.7.1 (Mar 27, 2017)

- Changed multiline parsing behavior to split input fields on escaped newlines if proper newlines are not found in the value. This should allow specifying escaped newline delimiters in non-textarea parameter inputs such as when using the Extended Choice Parameter Plugin.

Version 1.7 (Mar 1, 2017)

- Added support for 'MergePreserve' AC Handling Mode and new default option to 'Defer to Package' to override package AC Handling in Deploy CRX Packages Step.
- Added three new options for Validate CRX Content Packages step: "Forbidden AC Handling Modes", "Forbidden Filter Root Prefixes", and "Paths Denied for Inclusion".

Version 1.6.3 (Feb 28, 2017)

- Fixed credentials resolution AssertionError with Package Choice Parameter Definition.

Version 1.6.2 (Oct 26, 2016)

- I should not have removed the methods responsible for saving and loading the global config. pffft.

Version 1.6.1 (Oct 25, 2016)

- Removed a debug ERROR message from the build log.

Version 1.6 (Oct 25, 2016)

- Fixed the serialization of global plugin configuration parameters in master-slave Jenkins installations.
- Enhanced the "Preempt Login Patterns" global field to accept regular expressions.

Version 1.5.4 (Oct 24, 2016)

- Bumped granite-client-packman dependency version to 0.7.9 to bring compatibility with async-http-client:1.9.40.
- Added the ability to selectively enable preemptive Basic authentication via "Preempt Login for Base URLs" centralized plugin parameter.
- Removed validation-specific http client timeout parameters, which were made obsolete by switching to button-triggered validation on the build step configs.

Version 1.5.3 (Sep 29, 2016)

- Bumped granite-client-packman dependency version to 0.7.8 to change the service availability check behavior. Instead of sending a GET to exec.json and expecting a 405, which is filtered or transformed in some user environments, the client now sends a cmd=delete request to a non-existing package id ("adamcin:no-such-package:1.0.0") along with a form parameter that is ignored by the package manager service but which will fail the request if it is handled by the SlingPostServlet as a node creation request (-F"jcr:primaryType=adamcin:NoSuchType"). This bogus POST parameter is now included on all requests to the package manager service to ensure that requests are not treated as successful node creation requests if the package manager service is down for whatever reason.
- Added logic to trim superfluous base URL elements from the end, so that if the configured Base URL ends with /, or /crx/packman/service.jsp, or /crx/packman, or /crx these elements are removed before constructing the API client.

Version 1.5.2 (Sep 28, 2016)

- Changed bouncycastle-api maven dependency from range [1.0.3,) to explicit 1.0.3 instead because the latest 2.16.0 release was getting pulled in on install to a 1.610 jenkins core, which is not a good thing.
- Converted uses of single httpsig Key instances to build entire Keychain instead

Version 1.5.1 (Sep 27, 2016)

- Re-added bouncycastle dependency via bouncycastle-api plugin that was removed in an older pull request to fix an issue where Credentials selection failed when SSH Private Key entries were among the options
- Replaced each instance of automatic Base URL validation with a Test Connection button in the Advanced section
- Code cleanup

Version 1.5 (Sep 24, 2016)

- Upgraded Jenkins plugin dependencies for better master-slave behavior
- Resolved several issues related to serialization in master-slave configurations

Version 1.4 (Sep 19, 2016)

- Upgraded dependencies and plugin parent pom to support Jenkins 2.0
- Various bug fixes

Version 1.3.2 (August 27, 2014)

- Added similar sanity checks to prevent NPE's from other AEM 6 Oak responses which do not specify a charset.

Version 1.3.1 (August 27, 2014)

- Added a sanity check to prevent an NPE when a simple response from AEM 6 Oak does not specify a charset.

Version 1.3 (July 24, 2014)

- Added ability to override a component's configured credentials with the User Info portion of the base URI, where username and password are separated by a colon and followed by the 'at' ('@') symbol, and prepended to the hostname.
- Cleaned up the online help files.

Version 1.2.2 (July 20, 2014)

- Fixed out-of-memory errors when downloading large packages.
- Moved Base URL fields to the top of config forms, out from the hidden Connection Properties... sections.
- Changed the order of Existing Package Behavior options, which effectively changes the implicit default from "Uninstall and Delete" to "Overwrite".

Version 1.2.1 (Apr 4, 2014)

- Fixed severe memory leak due to improper creation and recycling of AsyncHttpClient instances.

Version 1.2 (Mar 20, 2014)

- Added "Build a Content Package on CRX" build step
- Added "Validate CRX Content Packages" build step
- Added support for globbing patterns in the Deploy and Validate steps' Package ID field, as an alternative to Package ID filters.
- Added a Wait Delay field for the Deploy step to help address bundle installation race conditions.

Version 1.1 (Mar 13, 2014)

- Added "Replicate on Install" option to "Deploy Content Packages to CRX" build step
- Added "Replicate Content Packages from CRX" build step
- Added validation-specific timeout options to the "CRX Content Package Deployer - HTTP Client" global options
- Fixed a bug in the granite-client-packman library where reserved and illegal characters in Package Ids were not being escaped when constructing URLs.

Version 1.0 (Jan 5, 2014)

- Initial release