CreateReleaseBundle

Overview

The **CreateReleaseBundle** is a native step that produces a Release Bundle for distribution to an Artifactory Edge Node. The step can be used to create a signed or unsigned release bundle. When the `sign` tag is set as `true` in your yaml, this step saves artifact information to support signed pipelines.

Page Contents

YAML Schema

The YAML schema for DockerBuild native step is as follows:

CreateReleaseBundle

```
pipelines:
  - name:    <string>
    steps:
      - name: <string>
        type: CreateReleaseBundle
        configuration:
          #inherits all the tags from bash; https://www.jfrog.com/confluence/display/JFROG/Bash
          releaseBundleName:        <string>
          releaseBundleVersion:     <string>
          dryRun:                   <boolean>   # default true
          sign:                     <boolean>   # default false
          description:              <string>    # optional
          failOnValidate:           <boolean>   # optional (Signed Pipelines must be enabled)
          releaseNotes:                         # optional
            content:                <string>    # "markdown|asciidoc|plain_text"
            syntax:                 <string>    # required in releaseNotes

          inputResources:
            - name:                 <BuildInfo resource>  # one or more BuildInfo, or
            - name:                 <Aql resource>        # one Aql
          outputResources:
            - name:                 <ReleaseBundle resource>

        execution:
          onStart:
            - echo "Preparing for work..."
          onSuccess:
            - echo "Job well done!"
          onFailure:
            - echo "uh oh, something went wrong"
          onComplete: #always
            - echo "Cleaning up some stuff"
```

Tags

name

An alphanumeric string (underscores are permitted) that identifies the step.

type

Must be `CreateReleaseBundle` for this step type.

configuration

Specifies all configuration selections for the step's execution environment. This step inherits the Bash/PowerShell step configuration tags, including these pertinent tags:

Tag	Description of usage	Required /Optional
`inputResources`	Must specify either a named BuildInfo resource(s) <u>or</u> an Aql resource. CreateReleaseBundle step does not accept other input resources. If BuildInfo `inputResources` are provided, the query for the release bundle is constructed using the `buildName`, `buildNumber`, and `targetRepo` of each BuildInfo input.	Required
`outputResources`	Must specify a ReleaseBundle resource. The `name`, `version`, and `isSigned` settings in the output ReleaseBundle are updated to the step's `releaseBundleName`, `releaseBundleVersion`, and `sign` values respectively (or any environment variable values that replaced environment variable placeholders for those values).	Required

In addition, these tags can be defined to support the step's native operation:

All native steps derive from the Bash step. This means that all steps share the same base set of tags from Bash, while native steps have their own additional tags as well that support the step's particular function. So it's important to be familiar with the Bash step definition, since it's the core of the definition of all other steps.

Tag	Description of usage	Required /Optional
`releaseBundleName`	An alphanumeric name for the release bundle.	Required
`releaseBundleVersion`	Version string for the release bundle.	Required
`dryRun`	When set to `true` parse and validate only to test whether a release bundle version can be created. Default is true.	Optional
`sign`	Specifies whether the release bundle version will be signed as part of this step. Default is false. **GPG Signing Key Passphrase** When configuring `sign=true`, and when your Distribution release bundle GPG signing key is passphrase protected, remember to provide the GPG Signing Key Passphrase when creating /updating your Distribution Integration .	Optional
`description`	Description of the release bundle.	Optional
`releaseNotes`	Describes the release notes for the release bundle version. `syntax` specifies the format of release notes: `plain_text`, `markdown`, or `asciidoc`. Default is `plain_text`. `content` is the release notes string in the specified syntax format. Use the `|` character to denote a string preserving newlines.	Optional

`failOnValidate`	Fail the step if one of the signatures of the BuildInfo input resource artifacts cannot be verified. Default is `false`.	Optional (Signed Pipelines must be enabled)

execution

Declares collections of shell command sequences to perform for pre- and post-execution phases:

Tag	Description of usage	Required/Optional
`onStart`	Commands to execute in advance of the native operation	Optional
`onSuccess`	Commands to execute on successful completion	Optional
`onFailure`	Commands to execute on failed completion	Optional
`onComplete`	Commands to execute on any completion	Optional

The actions performed for the `onExecute` phase are inherent to this step type and may not be overridden.

Examples

The following examples show which settings to configure for a few different release bundles.

Unsigned Release Bundle Created using BuildInfo Resource

A simple, unsigned release bundle created using a BuildInfo resource. In this case, the release bundle version will be the run number and will have no description or release notes.

- This example requires an Artifactory Integration and a Distribution Integration.
- The Pipelines DSL for this example is available in this repository in the JFrog GitHub account.

CreateReleaseBundle

```
template: true   # required for local templates
valuesFilePath: ./values.yml

resources:
  # Build info of first build to bundle
  - name: gosvc_promoted_build_info
    type: BuildInfo
    configuration:
      sourceArtifactory: {{ .Values.myArtifactoryIntegration }}
      buildName: svc_build
      buildNumber: 1

  # Build info of second build to bundle
  - name: appl_promoted_build_info
    type: BuildInfo
    configuration:
      sourceArtifactory: {{ .Values.demoArtifactoryIntegration }}
      buildName: backend_build
      buildNumber: 1

  # Release bundle
  - name: release_bundle
    type: ReleaseBundle
    configuration:
      sourceDistribution: {{ .Values.distributionIntegration }}
      name: demo_rb
      version: v1.0.0

  # Signed version of the same release bundle
  - name: signed_bundle
    type: ReleaseBundle
    configuration:
      sourceDistribution: {{ .Values.distributionIntegration }}
      name: demo_rb
      version: v1.0.0

  # Distribution rules
  - name: distribution_rules
    type: DistributionRule
    configuration:
      sourceDistribution: {{ .Values.distributionIntegration }}
      serviceName: "*"
      siteName: "*"
      cityName: "*"
      countryCodes:
        - "CN"
        - "GB"

pipelines:
  - name: demo_release_mgmt
    steps:
      - name: bundle
        type: CreateReleaseBundle
        configuration:
          releaseBundleName: demo_rb
          releaseBundleVersion: v1.0.${run_number}
          dryRun: false
          sign: false
          description: "some random test description"
          inputResources:
            - name: gosvc_promoted_build_info
              trigger: true
            - name: appl_promoted_build_info
              trigger: true
          outputResources:
            - name: release_bundle
          releaseNotes:
            syntax: markdown
            content: |
              ## Heading
                * Bullet
                * Points

      - name: sign
        type: SignReleaseBundle
        configuration:
          inputResources:
            - name: release_bundle
          outputResources:
            - name: signed_bundle

      - name: distribute
        type: DistributeReleaseBundle
        configuration:
          dryRun: false
          inputResources:
            - name: signed_bundle
            - name: distribution_rules
```

Create and Sign Release Bundle

Create and immediately sign a release bundle, with a description and release notes added to the release bundle.

CreateReleaseBundle

```
pipelines:
  - name: createReleaseBundlePipeline
    steps:
      - name: createReleaseBundleStep
        type: CreateReleaseBundle
        configuration:
          releaseBundleName: myReleaseBundle
          releaseBundleVersion: "${run_number}"
          dryRun: false
          sign: true
          description: "My release bundle"
          releaseNotes:
            syntax: plain_text
            content: "Release of ${run_number} by ${step_name}"
          inputResources:
            - name: myBuildInfo
          outputResources:
            - name: myReleaseBundle
```

Trigger a Dry Run

Trigger a dry run of the release bundle creation. No release bundle will be created.

CreateReleaseBundle

```
pipelines:
  - name: createReleaseBundlePipeline
    steps:
      - name: createReleaseBundleStep
        type: CreateReleaseBundle
        configuration:
          releaseBundleName: myReleaseBundle
          releaseBundleVersion: "${run_number}"
          dryRun: true
          inputResources:
            - name: myBuildInfo
          outputResources:
            - name: myReleaseBundle
```

How it Works

When you use the **CreateReleaseBundle** native step in a pipeline, it performs the following functions in the background:

- jfrog rt config (configure cli with the integration in the input resource)
- jfrog rt curl (get the Artifactory service_id)
- validate_artifact (use the signed pipelines feature to verify the incoming BuildInfo)
- Create the release bundle creation payload (the JSON object that will be in the request to Distribution)
- curl (send the release bundle creation payload to Distribution)
- write_output (update the output ReleaseBundle resource)
- save_artifact_info (if the bundle was signed, save the data for use with signed pipelines)