JENKINS CODE SIGNING

Integration Guide

Applicable Devices:

KMES Series 3

TABLE OF CONTENTS

[1] OVERVIEW

This section will answer the following two questions:

- What is a Jenkins plugin (in our own words)?
- What is the purpose of the FXCL Jenkins Plugin?

[1.1] WHAT IS A JENKINS PLUGIN (IN OUR OWN WORDS)?

A ***plugin*** implements one or more build steps for consumption by a pipeline or project. By convention, though, a plugin generally implements only one build step for the purpose of separation.

A ***pipeline*** is essentially a set of configurable build-steps. This terminology generally refers to the "Pipeline" plugin in general.

A ***build step*** is the fundamental building block of build automation in Jenkins. Everything that does an action in your project is a "build step". Everything from pulling from git to notifying users that a build is complete is a build step.

In relation, a ***stage*** is a logical grouping of those build steps. For example, you can "skip the Test stage", whereas otherwise, you'd need to specify individual build steps to skip over. It can also help visualize progress for consumers of the build results. Stages are not a required part of a pipeline and are strictly logical, meaning that it's not necessary to worry about stages other than understanding what a stage is.

Pipeline scripting comes in two forms:

- **Declarative** - A pre-defined set of tasks and environments to run build steps and/or Groovy expressions in.

- Imperative (or **Scripted**) - Similar to the declarative version but with some limitations because of the lack of a declarative environment. In exchange, you obtain the full power of Groovy. It's very powerful, but difficult to use.

[1.2] WHAT IS THE PURPOSE OF THE FXCL JENKINS PLUGIN?

The problem with the existing Jenkins code signing plugins is that they have no notion of an "approval" process. The resulting signature must be given immediately, or failure occurs. When a request to sign is submitted, it will fail because there is no time for approval to occur. There's also no way to query for the same request using something like Jarsigner, which relies on PKCS11. Thus, the need for a plugin that handles the approval process arises.

The FXCL Jenkins Plugin accomplishes the following:

- By interfacing with the KMES Series 3 registration authority, it allows for the standard approval process to take place.
- By incorporating Futurex Client Library (FXCL) functionality into the plugin, it makes it possible to sign files in bulk.

[2] PREREQUISITES

Before deploying code signing capabilities, a range of prerequisites must be met on the KMES Series 3 and on the computer that will be running the Jenkins instance.

[2.1] KMES-SPECIFIC PREREQUISITES

- Initial setup and configuration of the KMES Series 3, including loading a Platform Master Key and network setup.
- Host API port (2001 by default) unblocked on any firewalls the KMES Series 3 is behind.

[2.2] JENKINS-SPECIFIC PREREQUISITES

- In this integration guide we'll be using the Web application ARchive (WAR) file version of Jenkins. It can be installed on any operating system or platform that runs a version of Java supported by Jenkins. See the Java Requirements page on the jenkins.io website for details.
- The FXCL Jenkins Plugin file needs to be downloaded from the Futurex Portal to the computer that will be running the Jenkins instance.
- Minimum hardware requirements:
  - 256 MB of RAM
  - 1 GB of drive space

  Please refer to Hardware Recommendations page on the jenkins.io website for a comprehensive list of hardware recommendations.
- Software requirements:
  - Java: see the Java Requirements page on the jenkins.io website
  - Web browser: see the Web Browser Compatibility page on the jenkins.io website
  - For Windows operating system: Windows Support Policy

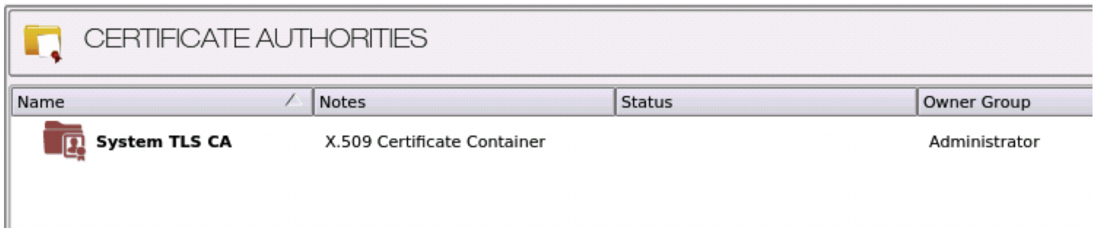
[3] KMES SERIES 3 CONFIGURATION

The first half of this section covers the steps needed to configure TLS communication between the KMES Series 3 and the Jenkins instance. The second half of this section covers general KMES configurations that must be made for the KMES to provide Jenkins code signing functionality.

[3.1] CONFIGURE TLS COMMUNICATION BETWEEN THE KMES SERIES 3 AND THE JENKINS INSTANCE

[3.1.1] Create a Certificate Authority (CA)

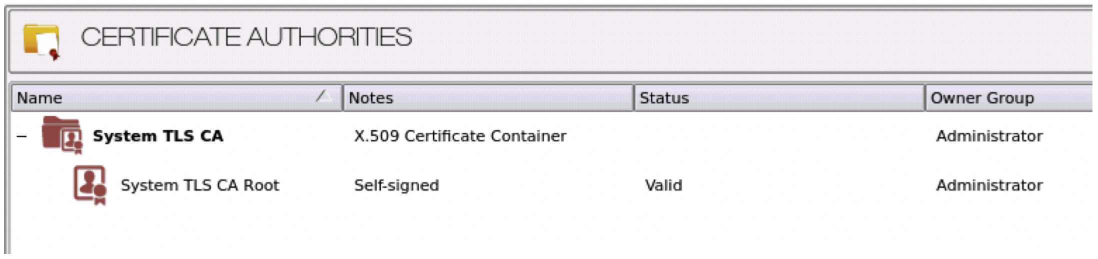
1. Log in to the KMES Series 3 application interface with the default Admin identities.
2. Select **PKI** > **Certificate Authorities** in the left menu, then click the **[Add CA...]** button at the bottom of the page.
3. In the **Certificate Authority** dialog, enter a name for the Certificate Container, leave all other fields as the default values, then click **[OK]**.
4. The Certificate Container that was just created will be listed now in the Certificate Authorities menu.



CERTIFICATE AUTHORITIES

Name	Notes	Status	Owner Group
System TLS CA	X.509 Certificate Container		Administrator

5. Right-click on the Certificate Container and select **Add Certificate** > **New Certificate...**
6. In the **Subject DN** tab, set a Common Name for the certificate, such as "System TLS CA Root".
7. In the **Basic Info** tab, leave the default settings.
8. In the **V3 Extensions** tab, select the **Certificate Authority** profile, then click **[OK]**.
9. The root CA certificate will be listed now under the previously created Certificate Container.

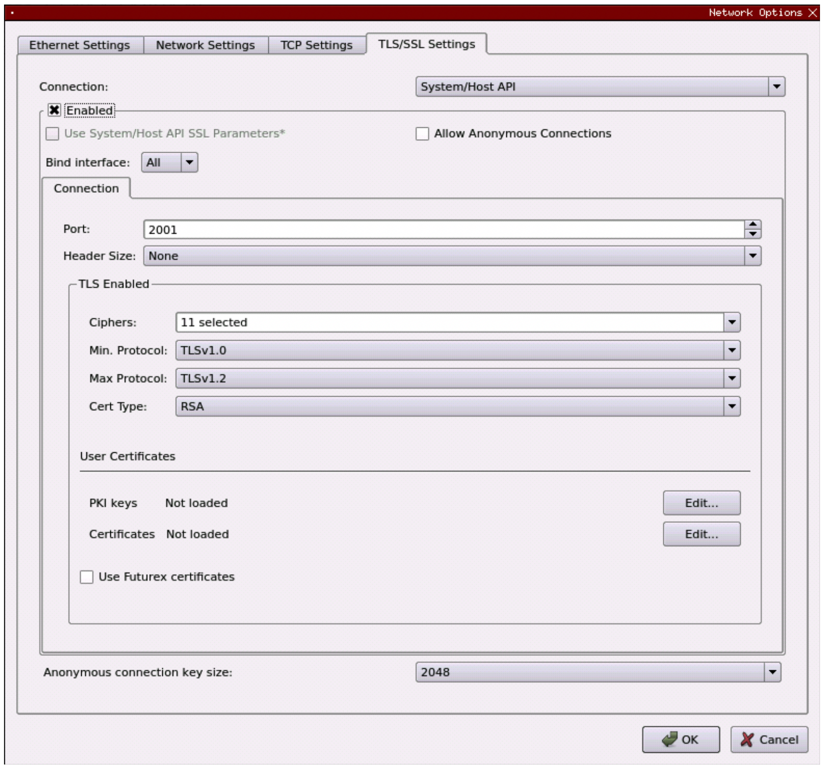


CERTIFICATE AUTHORITIES

Name	Notes	Status	Owner Group
– System TLS CA	X.509 Certificate Container		Administrator
System TLS CA Root	Self-signed	Valid	Administrator

[3.1.2] Generate a CSR for the System/Host API connection pair

1. Go to **Administration** > **Configuration** > **Network Options**.
2. In the **Network Options** dialog, select the **TLS/SSL Settings** tab.
3. Under the **System/Host API** connection pair, uncheck **Use Futurex certificates**, then click the **[Edit...]** button next to PKI keys in the User Certificates section.

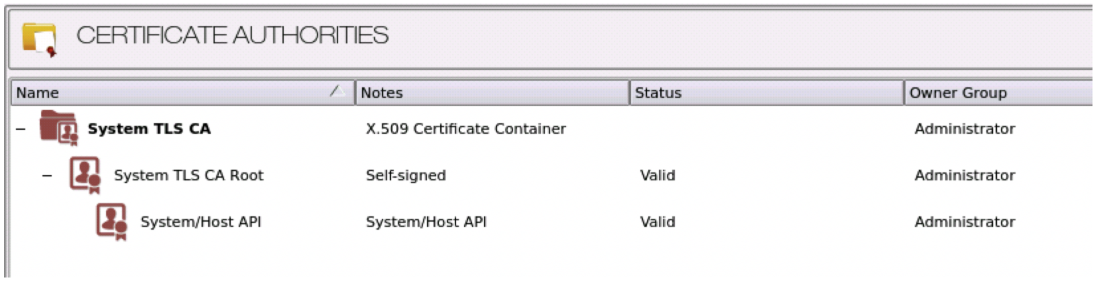


Connection: System/Host API
[x] Enabled
[ ] Use System/Host API SSL Parameters*
[ ] Allow Anonymous Connections
Bind interface: All
Connection
Port: 2001
Header Size: None
TLS Enabled
Ciphers: 11 selected
Min. Protocol: TLSv1.0
Max Protocol: TLSv1.2
Cert Type: RSA
User Certificates
PKI keys Not loaded Edit...
Certificates Not loaded Edit...
[ ] Use Futurex certificates
Anonymous connection key size: 2048

4. In the **Application Public Keys** dialog, click **[Generate...]**.
5. There will be a warning stating that SSL will not be functional until new certificates are imported. Select **[Yes]** if you wish to continue.
6. In the **PKI Parameters** dialog, leave the default settings and select **[OK]**.
7. It should show that a PKI Key Pair is loaded now in the **Application Public Keys** dialog. If this is the case, click **[Request...].**
8. In the **Subject DN** tab, you can leave the default **System/Host API** value set in the **Common Name** field, or you can change it to a different value.
9. In the **V3 Extensions** tab, select the **TLS Server Certificate** profile.
10. In the **PKCS #10 Info** tab, select a save location for the CSR, then click **[OK]**.
11. There should be a message stating that the certificate signing request was successfully written to the file location that was selected. Click **[OK]**.
12. Click **[OK]** again to save the **Application Public Keys** settings.
13. In the main **Network Options** dialog, it should now say **Loaded** next to **PKI keys**.

[3.1.3] Sign the System/Host API CSR

1. Go to the **PKI** > **Certificate Authorities** menu.
2. Right-click on the root CA certificate created in section 3.1.1, then select **Add Certificate** > **From Request...**
3. In the file browser, find and select the CSR that was generated for the System/Host API connection pair.
4. Once loaded, none of the settings need to be modified for the certificate. Click **[OK]**.
5. The signed System/Host API certificate should now show under the root CA certificate on the **Certificate Authorities** page.



CERTIFICATE AUTHORITIES

Name	Notes	Status	Owner Group
− System TLS CA	X.509 Certificate Container		Administrator
− System TLS CA Root	Self-signed	Valid	Administrator
System/Host API	System/Host API	Valid	Administrator

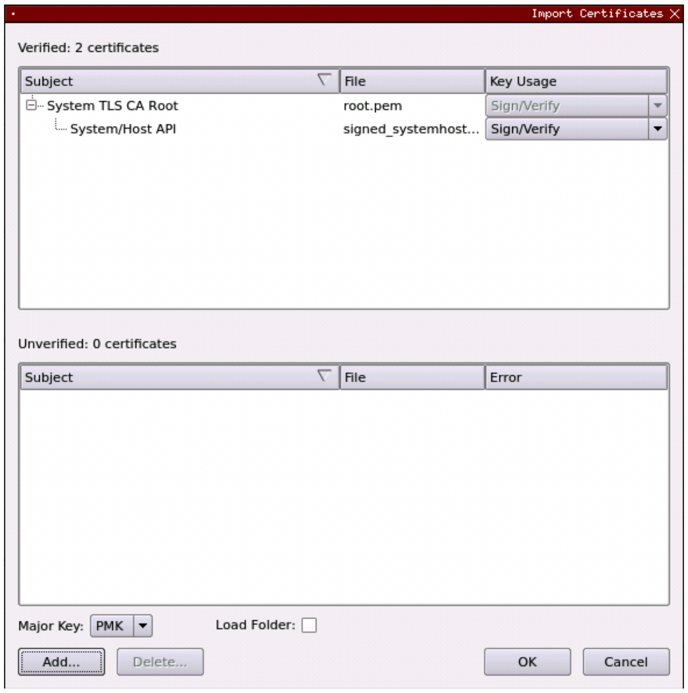
[3.1.4] Export the root CA and signed System/Host API certificates

1. Right-click on the root CA certificate, then select **Export** > **Certificate(s)...**
2. Change the encoding to **PEM**. Then click **[Browse...]** and specify a save location and name for the export file.
3. There should be a message stating that the file was successfully written to the location that was selected. Click **[OK]**.
4. Right-click on the signed System/Host API certificate, then select **Export** > **Certificate(s)...**
5. Change the encoding to **PEM**. Then click **[Browse...]** and select a save location and name for the export file.
6. There should be a message stating that the file was successfully written to the location that was selected. Click **[OK]**.

[3.1.5] Load the exported certificates into the System/Host API connection pair

1. Go to **Administration** > **Configuration** > **Network Options**.
2. In the **Network Options** dialog, select the **TLS/SSL Settings** tab.
3. Click **[Edit...]** next to Certificates in the User Certificates section.
4. Right-click on the **System/Host API SSL CA** X.509 Certificate Container, then select **Import...**
5. Click the **[Add...]** button at the bottom of the **Import Certificates** dialog.

6. In the file browser, find and select both the root CA certificate and the signed System/Host API certificate, then click **[Open]**. The certificate chain should appear as shown below:



Verified: 2 certificates

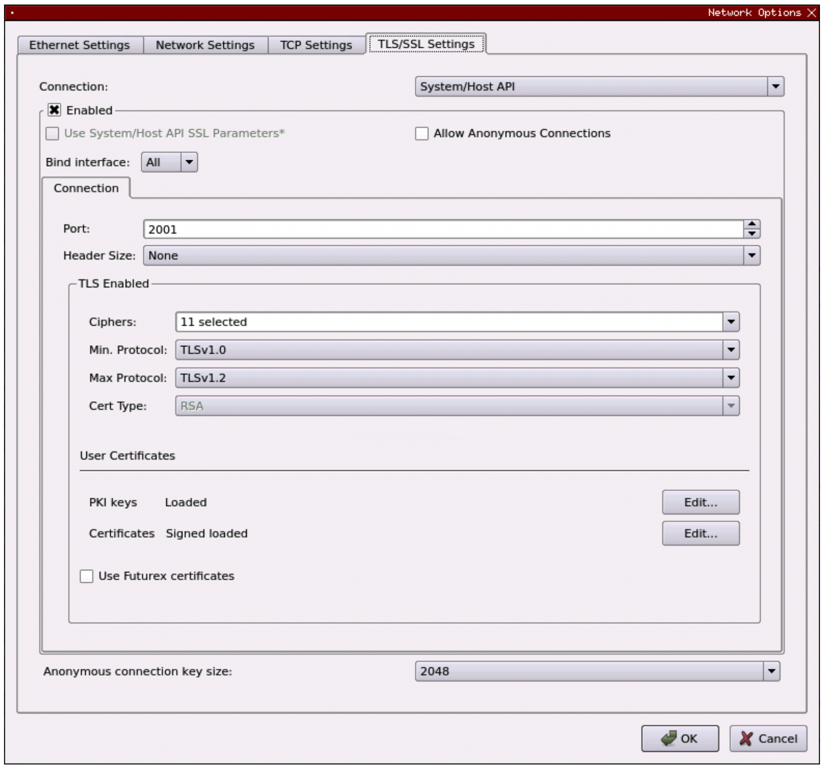
Subject	File	Key Usage
System TLS CA Root	root.pem	Sign/Verify
System/Host API	signed_systemhost...	Sign/Verify

Unverified: 0 certificates

Subject	File	Error

Major Key: PMK
Load Folder: ☐

7. Click **[OK]** to save the changes. In the **Network Options** dialog, the System/Host API connection pair should show **Signed loaded** next to Certificates in the User Certificates section, as shown below:

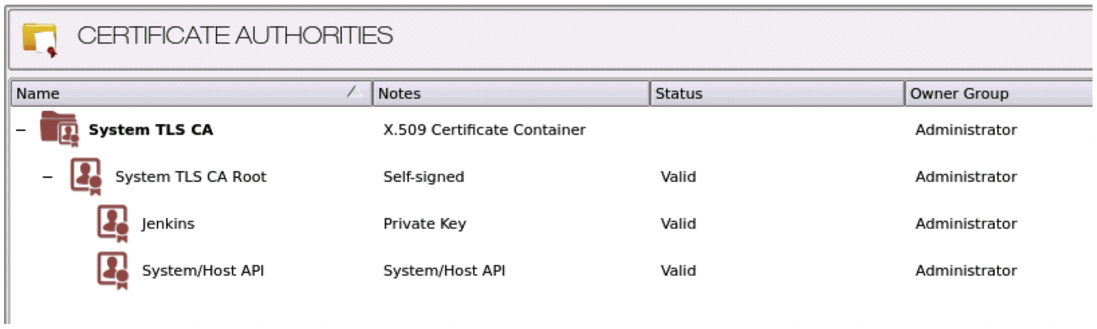


Connection: System/Host API
☒ Enabled
☐ Use System/Host API SSL Parameters*
☐ Allow Anonymous Connections
Bind interface: All
Port: 2001
Header Size: None
TLS Enabled
Ciphers: 11 selected
Min. Protocol: TLSv1.0
Max Protocol: TLSv1.2
Cert Type: RSA
User Certificates
PKI keys Loaded
Certificates Signed loaded
☐ Use Futurex certificates
Anonymous connection key size: 2048

8. Click **[OK]** to save and exit the Network Options dialog.

[3.1.6] Generate a signed certificate for the Jenkins instance

1. Go to the **PKI** > **Certificate Authorities** menu.
2. Right-click on the root CA certificate and select **Add Certificate** > **New Certificate...**
3. In the **Subject DN** tab, set a Common Name for the certificate, such as "Jenkins".
4. All settings in the **Basic Info** tab can be left as the default values.
5. In the **V3 Extensions** tab, select the **TLS Client Certificate** profile, then click **[OK]**.
6. The signed Jenkins certificate will be listed now under the root CA certificate.



Name	Notes	Status	Owner Group
System TLS CA	X.509 Certificate Container		Administrator
System TLS CA Root	Self-signed	Valid	Administrator
Jenkins	Private Key	Valid	Administrator
System/Host API	System/Host API	Valid	Administrator

[3.1.7] Allow export of certificates using passwords

1. Navigate to **Administration** > **Configuration** > **Options**.
2. Check the box next to the second menu option, which says, "Allow export of certificates using passwords".
3. Click **[Save]**.

[3.1.8] Export the signed Jenkins certificate as a PKCS #12 file

1. Go to the **PKI** > **Certificate Authorities** menu.
2. Right-click on the signed Jenkins certificate, then select **Export** > **PKCS12...**
3. Select the **[Set Password]** button and enter a password for the PKCS #12 file, then click **[Save]**.
4. In the **Export Certificate** dialog, select **Export Selected Certificate with Parents** under Export Options, then click **[Next]**.
5. Specify a name for the PKCS #12 export file and click **[Open]**.
6. A message should appear stating the PKCS#12 certificate export was successful.

Note: This PKCS #12 file needs to be moved to the computer running the Jenkins instance. In a later section, it will be imported in Jenkins and used for TLS communication with the KMES Series 3.

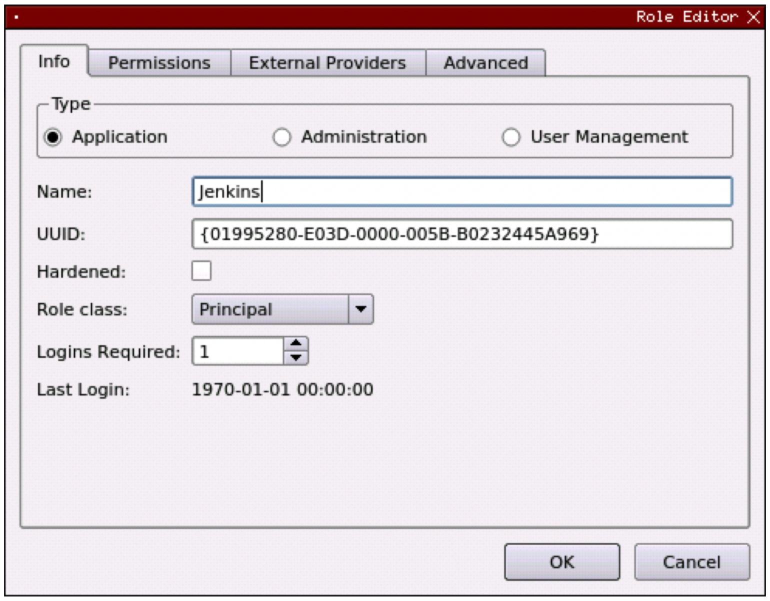
[3.2] GENERAL KMES CONFIGURATIONS FOR COMMUNICATION BETWEEN JENKINS AND THE KMES SERIES 3

[3.2.1] Enable the required Host API commands

1. Go to **Administration** > **Configuration** > **Host API Options**.
2. Enable the following commands:
    - **RAFA** - Enumerate issuance policies
    - **RAGA** - Retrieve issuance policy details
    - **RAGZ** - Retrieve Request (Authenticode)
    - **RAUZ** - Upload Request (Authenticode)
    - **RAGJ** - Retrieve Request (JAR)
    - **RAUJ** - Upload Request (JAR)
    - **RKLO** - Login User
    - **RAGO** - Retrieve Request (Hash Signing)
    - **RAUO** - Upload Request (Hash Signing)
3. Click **[Save]**.

[3.2.2] Create a Jenkins Role with the required permissions

1. Go to **Identity Management** > **Roles**, in the left menu, then click the **[Add...]** button at the bottom of the page.
2. Specify a name for the group, such as "Jenkins", then ensure that the settings below are selected.



Type: [✓ Application] Administration User Management

Name: Jenkins

UUID: {01995280-E03D-0000-005B-B0232445A969}

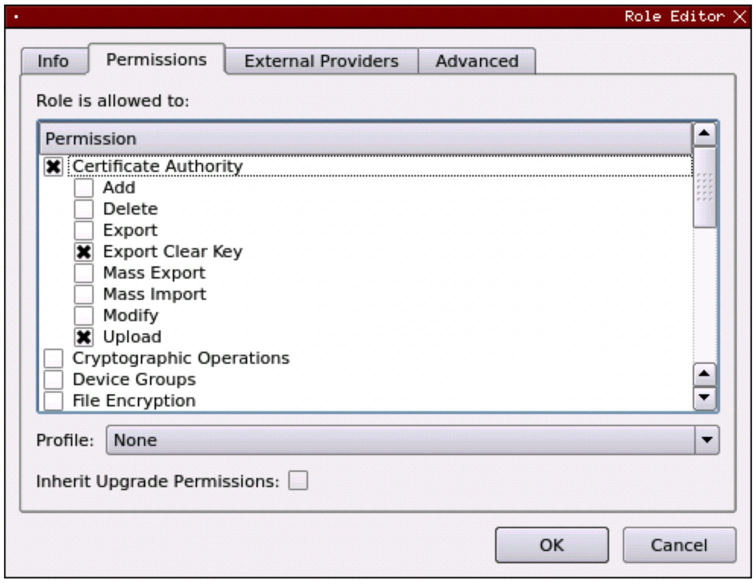
Hardened: ☐

Role class: Principal

Logins Required: 1

Last Login: 1970-01-01 00:00:00

3. In the **Permissions** tab, ensure that only the following **Certificate Authority** permissions are selected:



Role is allowed to:

Permission
- [✓] Certificate Authority
  - [ ] Add
  - [ ] Delete
  - [ ] Export
  - [✓] Export Clear Key
  - [ ] Mass Export
  - [ ] Mass Import
  - [ ] Modify
  - [✓] Upload
- [ ] Cryptographic Operations
- [ ] Device Groups
- [ ] File Encryption

Profile: None

Inherit Upgrade Permissions: [ ]

4. In the **Advanced** tab, ensure only **Host API** is selected for **Allowed Ports**.
5. Click **[OK]** to save.

[3.2.3] Create a Jenkins Identity with the correct assigned Roles

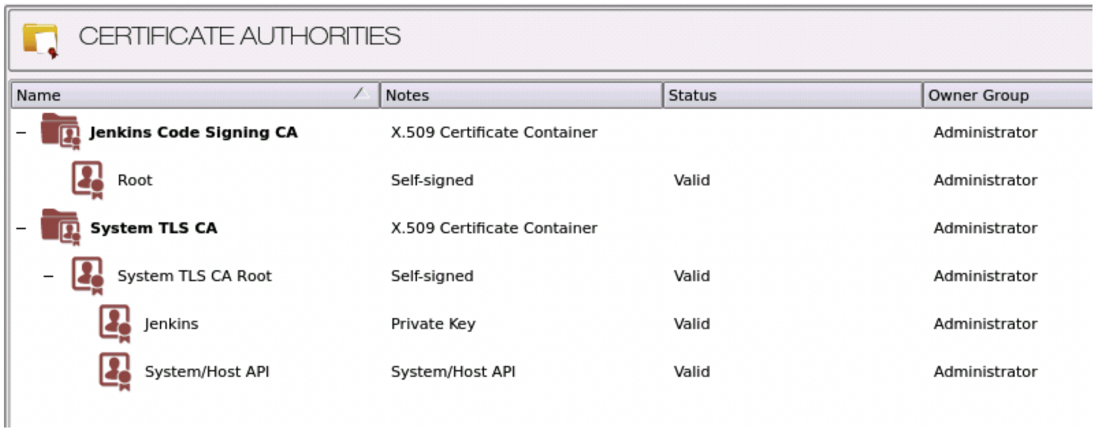
1. Navigate to the **Identity Management** > **Identities** menu, then right-click and select **Add** > **Client Application**.
2. In the **Info** tab, select **Application** for the Storage type and specify a name for the identity.
3. In the **Assigned Roles** tab, select the role you created in the previous section.
4. In the **Authentication** tab, remove the API Key mechanism, then add the password mechanism and set your password.
5. Click **[OK]** to finish creating the identity.

[3.2.4] Create a Signing Approval Group and give the Jenkins Role permissions to use it

1. Navigate to the **PKI** > **Signing Workflow** menu, then click the **[Add Approval Group...]** button at the bottom of the page.
2. Set a name for the Approval Group, such as "Jenkins", then click **[OK]** to save.
3. Right-click on the **Jenkins** Approval Group add select **Permission...**
4. Select the **Show all roles and permissions** checkbox, then grant the Jenkins role the **Use** permission and select **[OK]**.

[3.2.5] Create a Jenkins code signing certificate

1. Navigate to the **PKI** > **Certificate Authorities** menu, then click the **[Add CA...]** button at the bottom of the page.
2. In the **Certificate Authority** dialog, enter a name for the Certificate Container, such as "Jenkins Code Signing CA". Set the owner of the CA to the Jenkins role, then click **[OK]**.
3. The Certificate Container that was just created will be listed now in the Certificate Authorities menu.
4. Right-click on the Jenkins Certificate Container and select **Add Certificate** > **New Certificate...**
5. In the **Subject DN** tab, set a Common Name for the certificate, such as "Root".
6. In the **Basic Info** tab, leave the default settings.
7. In the **V3 Extensions** tab, select the **Code Signing Certificate** profile, then click **[OK]**.
8. The **Root** Jenkins code signing certificate will be listed now under the Jenkins Certificate Container.


CERTIFICATE AUTHORITIES

Name	Notes	Status	Owner Group
– **Jenkins Code Signing CA**	X.509 Certificate Container		Administrator
Root	Self-signed	Valid	Administrator
– **System TLS CA**	X.509 Certificate Container		Administrator
– System TLS CA Root	Self-signed	Valid	Administrator
Jenkins	Private Key	Valid	Administrator
System/Host API	System/Host API	Valid	Administrator

[3.2.6] Apply an Issuance Policy to the Jenkins code signing certificate

1. Go to the **PKI** > **Certificate Authorities** menu.
2. Right-click on the root certificate within the Jenkins Certificate Container, then select **Issuance Policy** > **Add...**
3. Under the **Basic Info** tab:
   - Specify an Alias if desired.
   - Set Approvals to **1** (**Note:** Setting Approvals to **0** will allow anonymous signing.)
   - Select any hashes that you wish to allow.
4. In the **X.509** tab, set the Default approval group to **Jenkins**.
5. In the **Object Signing** tab, select the **Allow object signing** checkbox.
6. Click **[OK]** to apply the Issuance Policy to the **Root** Jenkins code signing certificate.

[4] JENKINS DOWNLOAD, CONFIGURATION, AND FXCL JENKINS PLUGIN TESTING

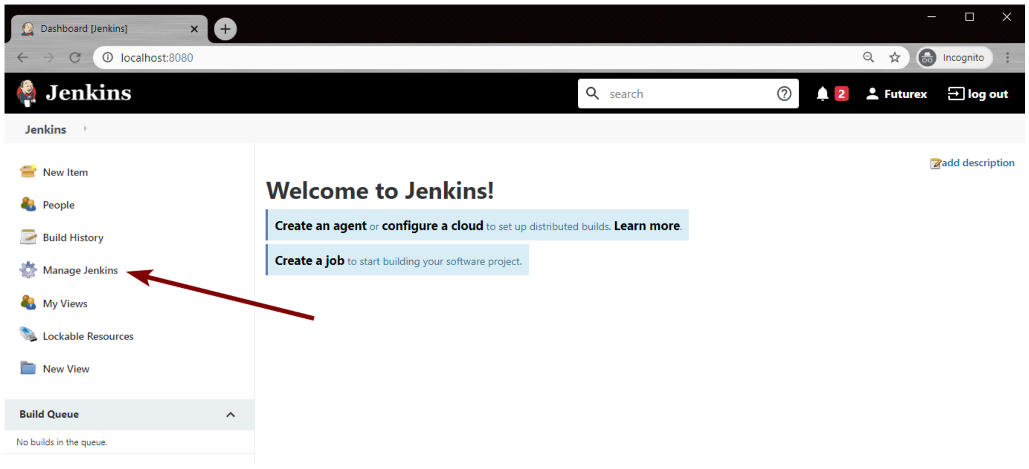
This section will cover the steps needed to download, run, and configure Jenkins so that the KMES Series 3 can be leveraged for code signing.

[4.1] DOWNLOADING, RUNNING, AND PERFORMING THE INITIAL JENKINS SETUP

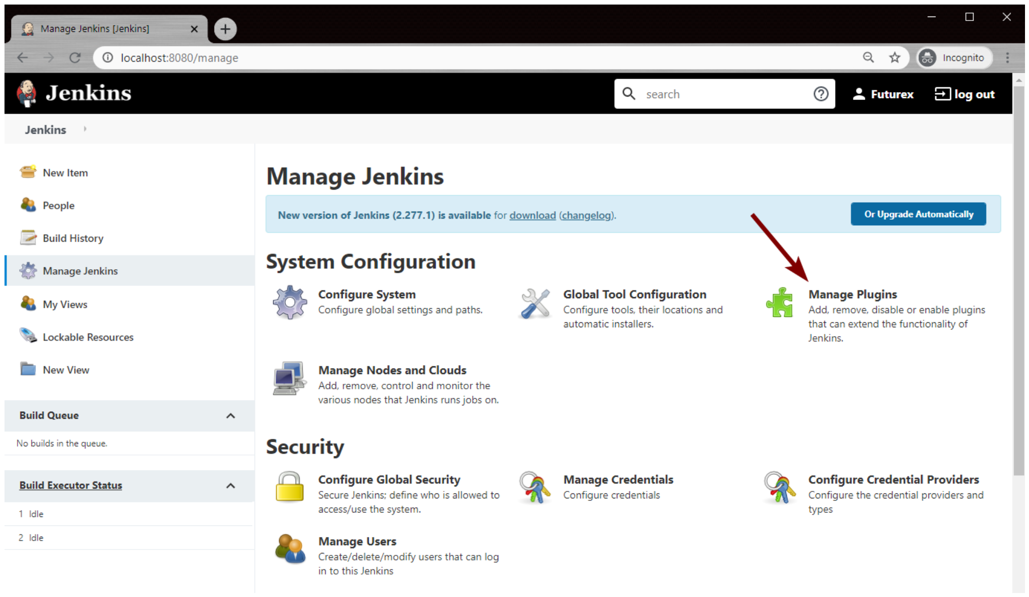
Download the jenkins.war file from https://www.jenkins.io/download/. Then follow the instructions for running the WAR file and completing the post-installation setup at the following url:https://www.jenkins.io/doc/book/installing/war-file/.

[4.2] INSTALLING THE FXCL JENKINS PLUGIN

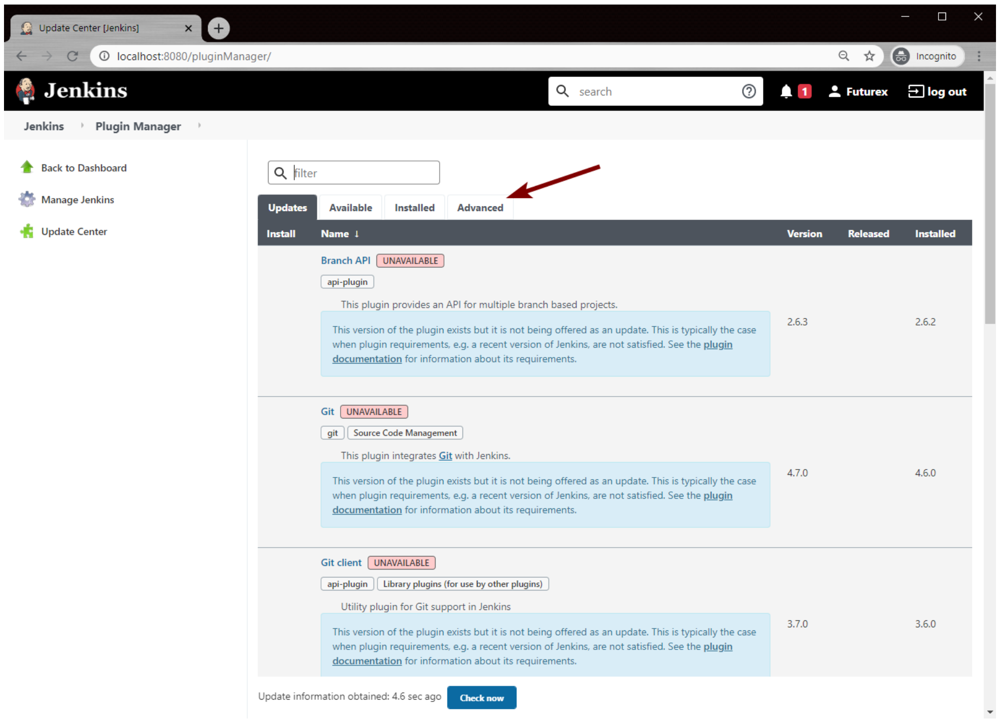
1. From the main Jenkins dashboard page, click the **Manage Jenkins** icon in the left-hand menu.



2. Click the **Manage Plugins** button in System Configuration section.


Manage Jenkins [Jenkins]
localhost:8080/manage
Incognito
Jenkins
search
Futurex
log out
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
New version of Jenkins (2.277.1) is available for download (changelog).
Or Upgrade Automatically
System Configuration
Configure System
Configure global settings and paths.
Global Tool Configuration
Configure tools, their locations and automatic installers.
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Security
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.
Manage Credentials
Configure credentials
Configure Credential Providers
Configure the credential providers and types
Manage Users
Create/delete/modify users that can log in to this Jenkins

3. On the Plugin Manager page, click the **Advanced** tab.



Update Center [Jenkins]

localhost:8080/pluginManager/

Jenkins | search | 1 | Futurex | log out

Jenkins › Plugin Manager ›

- Back to Dashboard
- Manage Jenkins
- Update Center

Filter

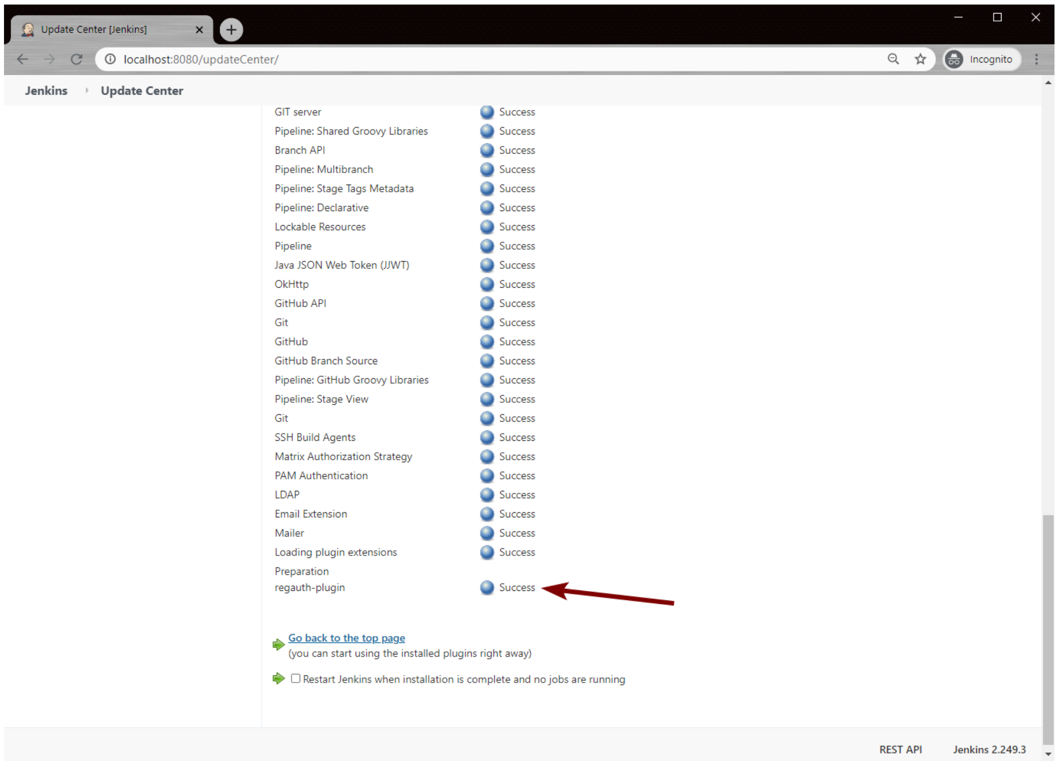
Updates | Available | Installed | Advanced

Install	Name ↓	Version	Released	Installed
	Branch API UNAVAILABLE api-plugin This plugin provides an API for multiple branch based projects. This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.	2.6.3		2.6.2
	Git UNAVAILABLE git Source Code Management This plugin integrates Git with Jenkins. This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.	4.7.0		4.6.0
	Git client UNAVAILABLE api-plugin Library plugins (for use by other plugins) Utility plugin for Git support in Jenkins This version of the plugin exists but it is not being offered as an update. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. See the plugin documentation for information about its requirements.	3.7.0		3.6.0

Update information obtained: 4.6 sec ago Check now

4. Scroll down to the Upload Plugin section and click the **Choose File** button. In the file browser, find and select the FXCL Jenkins Plugin file, then click **Upload**.

After clicking upload you will be redirected to the Update Center page where you can see the progress of the plugin installation. If the installation is successful the status of the FXCL Jenkins Plugin will change to "Success", as shown below:



Update Center [Jenkins]

localhost:8080/updateCenter/

Jenkins › Update Center

GIT server	Success
Pipeline: Shared Groovy Libraries	Success
Branch API	Success
Pipeline: Multibranch	Success
Pipeline: Stage Tags Metadata	Success
Pipeline: Declarative	Success
Lockable Resources	Success
Pipeline	Success
Java JSON Web Token (JJWT)	Success
OkHttp	Success
GitHub API	Success
Git	Success
GitHub	Success
GitHub Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Pipeline: Stage View	Success
Git	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Loading plugin extensions	Success
Preparation	
regauth-plugin	Success

Go back to the top page
(you can start using the installed plugins right away)

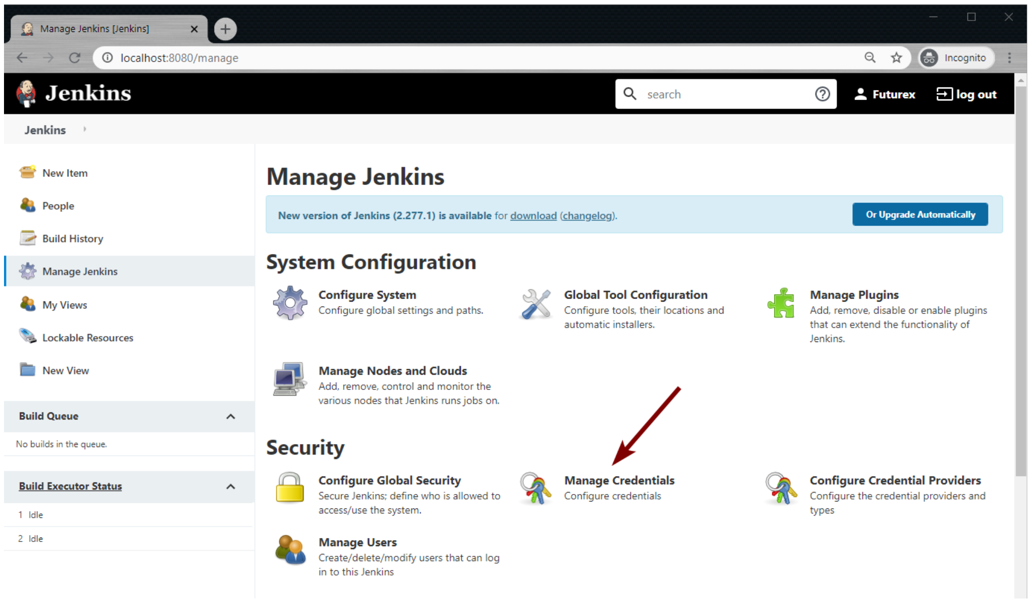
☐ Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.249.3

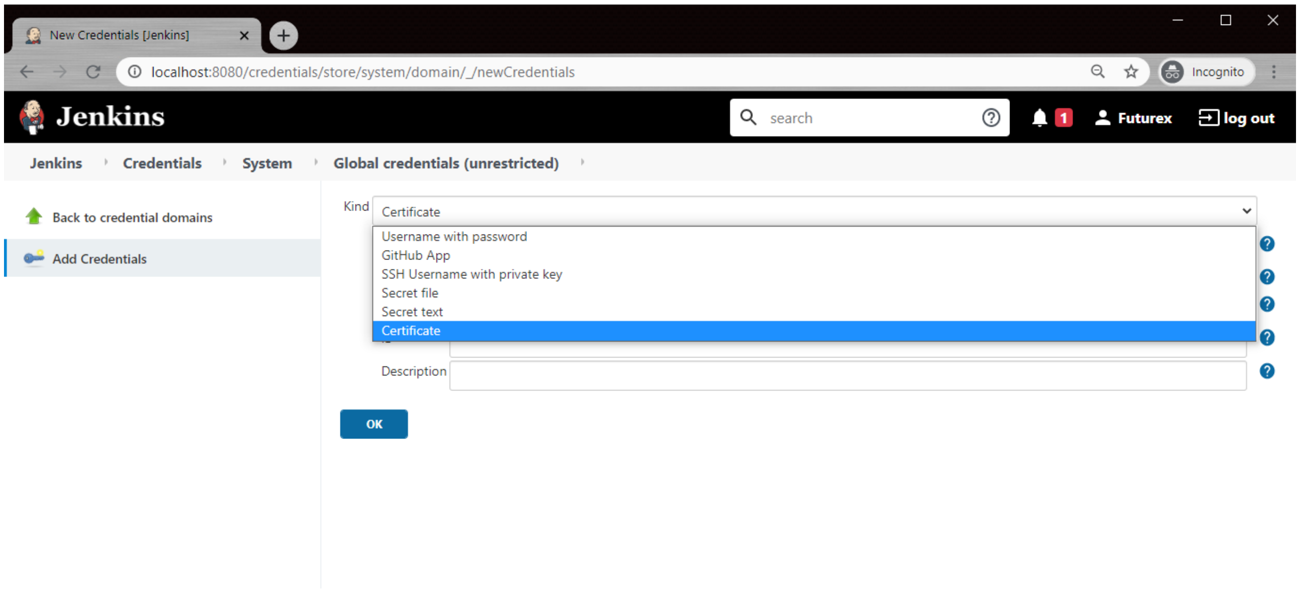
[4.3] REGISTER CERTIFICATE CREDENTIALS FOR TLS COMMUNICATION BETWEEN JENKINS AND THE KMES SERIES 3

In this section, the PKCS #12 file exported from the KMES in the "KMES Series 3 Configuration" section will be imported into Jenkins to be used for TLS communication. This PKCS #12 file contains the signed Jenkins certificate and the root (and intermediate certificate/s if applicable) certificate used to sign it, protected by a password.

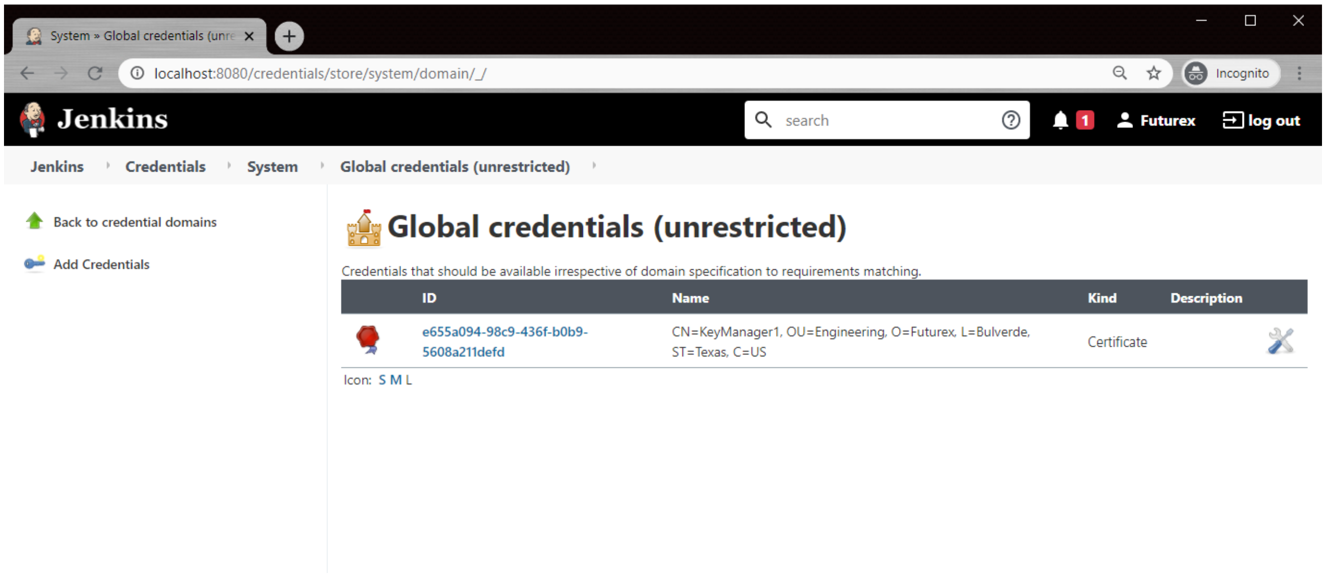
1. On the **Manage Jenkins** page, click the **Manage Credentials** button in the Security section.


Manage Jenkins [Jenkins]
localhost:8080/manage
Incognito
Jenkins
search
Futurex
log out
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
New version of Jenkins (2.277.1) is available for download (changelog).
Or Upgrade Automatically
System Configuration
Configure System
Configure global settings and paths.
Global Tool Configuration
Configure tools, their locations and automatic installers.
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Security
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.
Manage Credentials
Configure credentials
Configure Credential Providers
Configure the credential providers and types
Manage Users
Create/delete/modify users that can log in to this Jenkins

2. Select the **Jenkins** Store, contained within the global domain.
3. Select the **Global credentials (unrestricted)** Domain.
4. Click the **Add Credentials** button in the left-hand menu.
5. Change the value in the Kind dropdown to **Certificate**.


New Credentials [Jenkins]
localhost:8080/credentials/store/system/domain/_/newCredentials
Incognito
Jenkins
search
1
Futurex
log out
Jenkins
Credentials
System
Global credentials (unrestricted)
Back to credential domains
Add Credentials
Kind
Certificate
Username with password
GitHub App
SSH Username with private key
Secret file
Secret text
Certificate
Description
OK

6. Select the **Upload PKCS#12 certificate** radio button, then click **Choose File**. This will open the file browser. Find and select the .p12 file, then click **Open**. A message should appear that says, "Could retrieve key "system tls ca root". You may need to provide a password.
7. Click the **Change Password** button and enter the password of the PKCS #12 file.
8. Click the **OK** button to save the new credentials. They will now be listed on the following page:


System » Global credentials (unre
localhost:8080/credentials/store/system/domain/_/
Incognito
Jenkins
search
Futurex
log out
Jenkins › Credentials › System › Global credentials (unrestricted)
Back to credential domains
Add Credentials

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

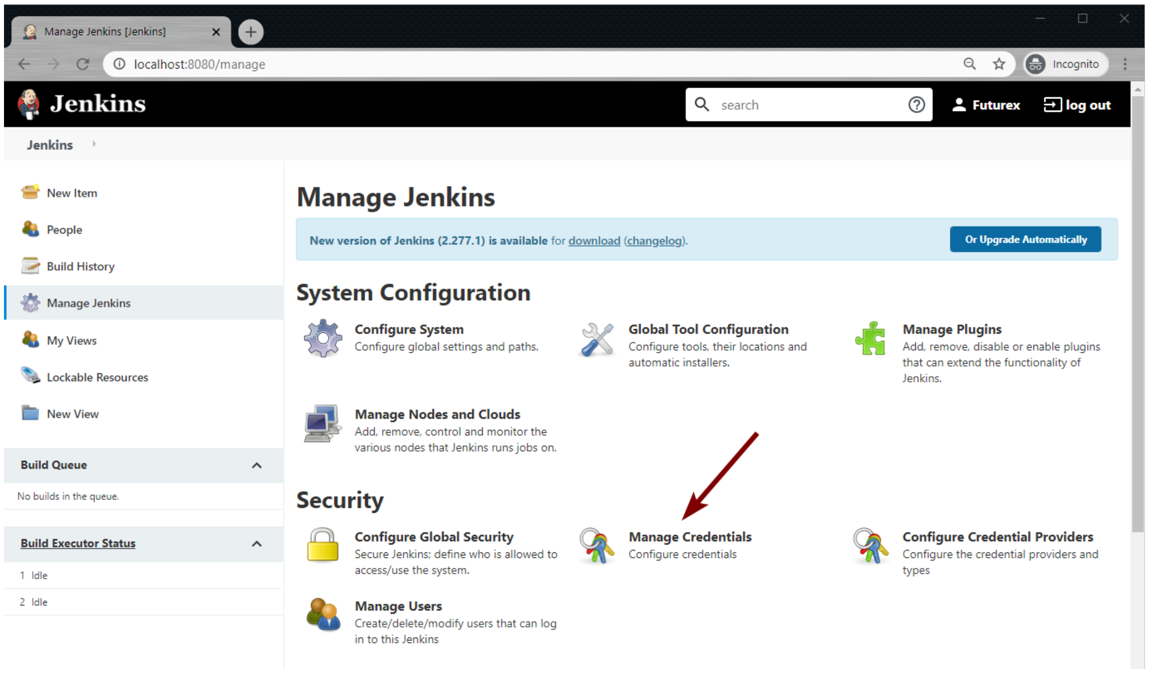
ID	Name	Kind	Description
e655a094-98c9-436f-b0b9-5608a211defd	CN=KeyManager1, OU=Engineering, O=Futurex, L=Bulverde, ST=Texas, C=US	Certificate	

Icon: S M L

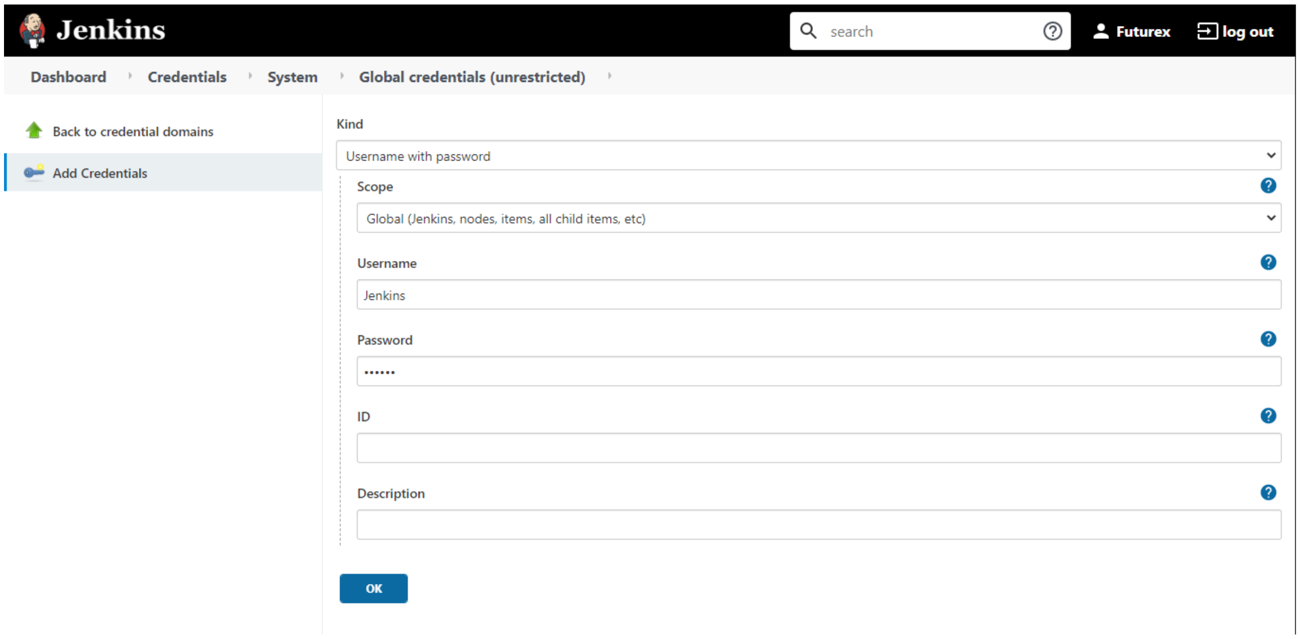
[4.4] REGISTER USERNAME WITH PASSWORD CREDENTIALS

In this section, username with password credentials will be configured in Jenkins for the "Jenkins" user that was created on the KMES in the "KMES Series 3 Configuration" section.

1. On the **Manage Jenkins** page, click the **Manage Credentials** button in the Security section.


Manage Jenkins [Jenkins]
localhost:8080/manage
Incognito
Jenkins
search
Futurex
log out
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
New version of Jenkins (2.277.1) is available for download (changelog).
Or Upgrade Automatically
System Configuration
Configure System
Configure global settings and paths.
Global Tool Configuration
Configure tools, their locations and automatic installers.
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Security
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.
Manage Credentials
Configure credentials
Configure Credential Providers
Configure the credential providers and types
Manage Users
Create/delete/modify users that can log in to this Jenkins

2. Select the **Jenkins** Store, contained within the global domain.
3. Select the **Global credentials (unrestricted)** Domain.
4. Click the **Add Credentials** button in the left-hand menu.
5. Leave the value in the Kind dropdown to the default value (i.e., **Username with password**).
6. In the **Username** and **Password** fields, specify the user name and password of the "Jenkins" user that was created on the KMES in the "KMES Series 3 Configuration" section.



Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: Jenkins

Password: ••••••

ID:

Description:

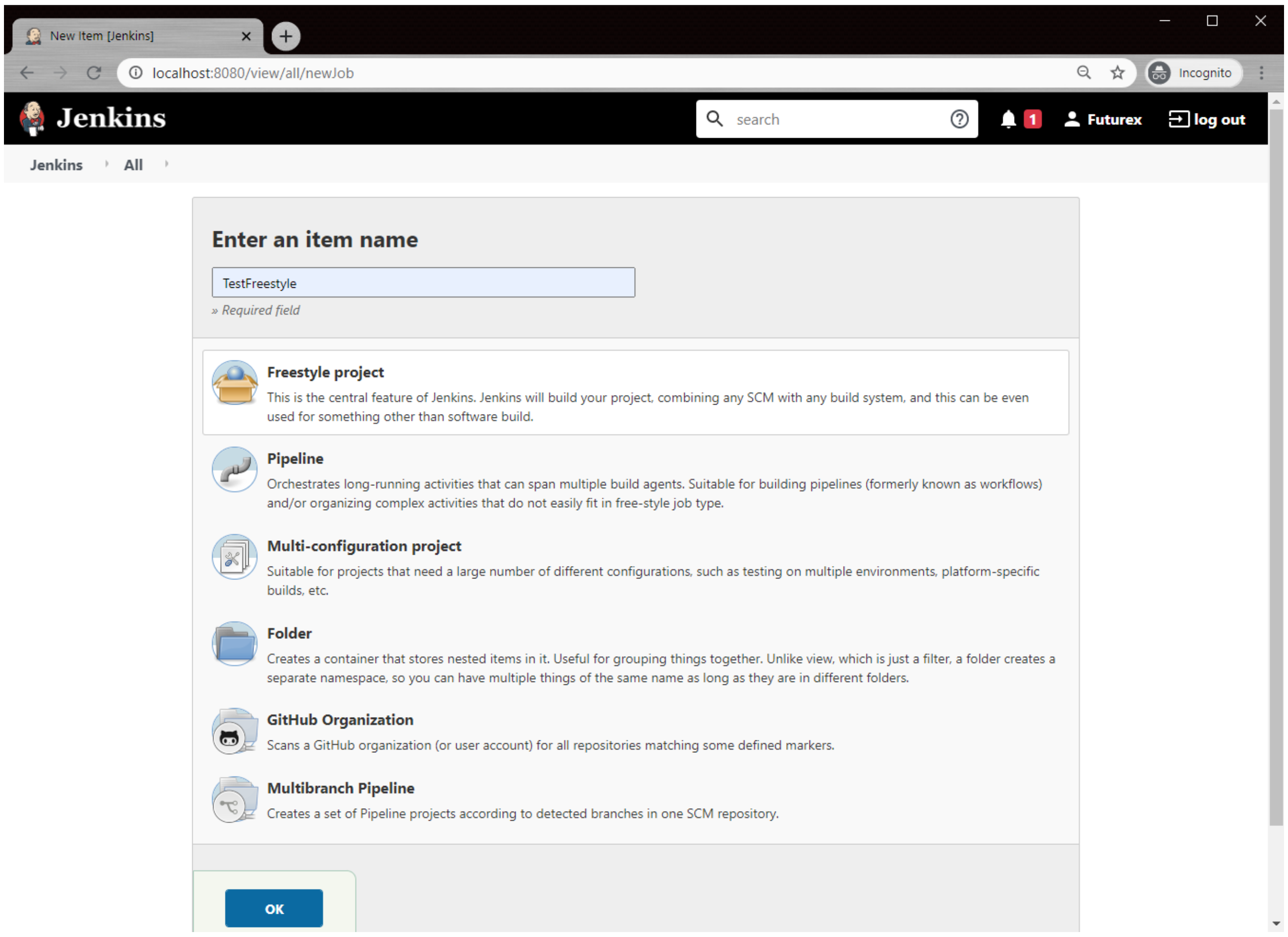
7. Click the **OK** button to save the new credentials.

[4.5] SIGNING A FILE IN A FREESTYLE PROJECT USING THE KMES SERIES 3 REGISTRATION AUTHORITY

This section will walk through creating, configuring, and running a new Freestyle project. If you want to use the KMES Series 3 registration authority to sign code in an existing Freestyle project, skip to step 6 in the next subsection.

[4.5.1] Creating and configuring a Freestyle project to leverage the KMES for code signing using the FXCL Jenkins Plugin

1. From the main Jenkins dashboard page, click the **New Item** icon in the left-hand menu.
2. Select **Freestyle project**, enter a name for the project, then click the **OK** button.



Enter an item name

TestFreestyle

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

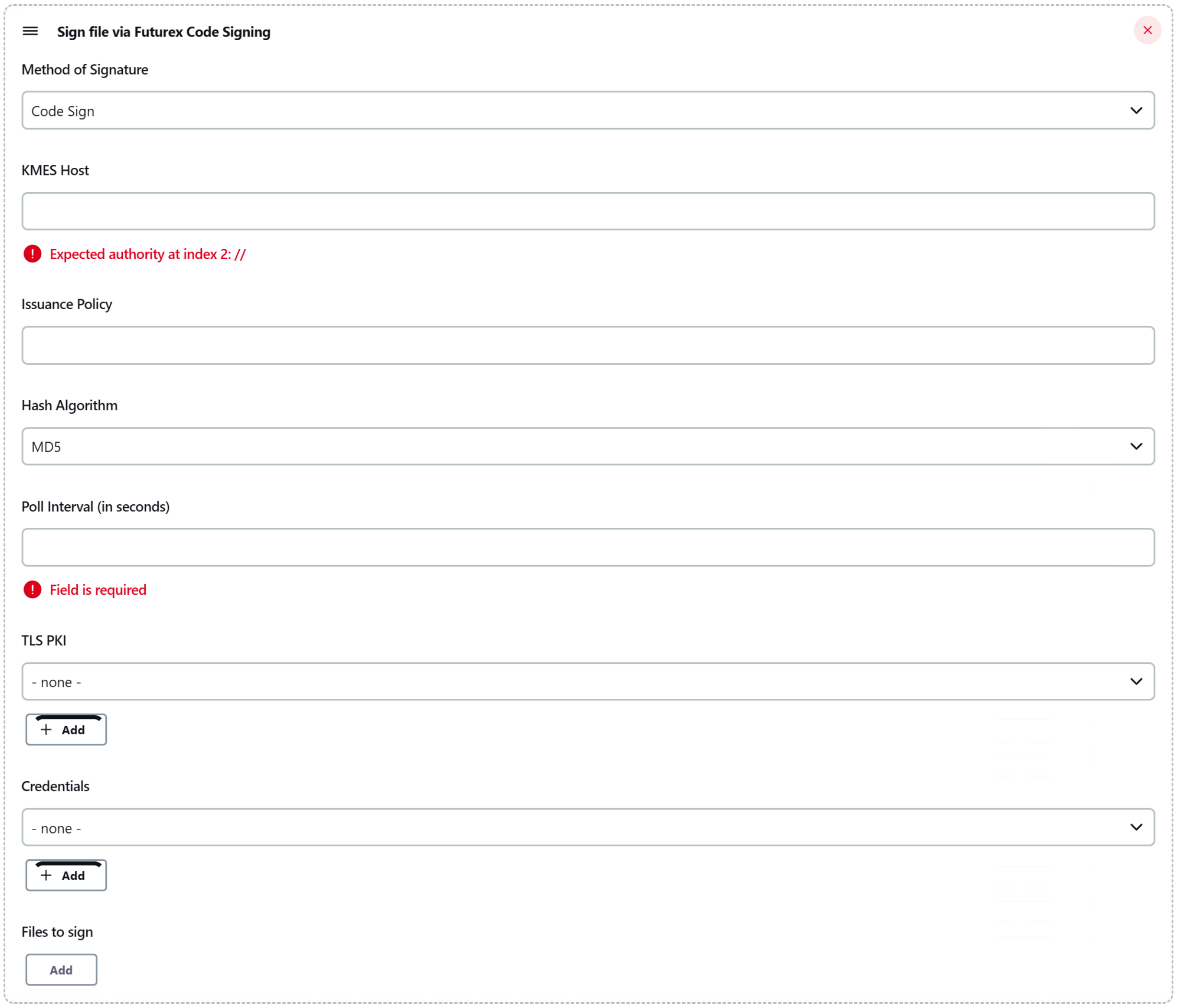
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

This will bring up the configuration page for the Freestyle project.

3. Scroll down to the **Build** section, click the **Add build step** button, and select **Sign file via Futurex Code Signing** in the dropdown. This option is provided by the FXCL Jenkins Plugin. The following box will appear:



Sign file via Futurex Code Signing

Method of Signature: Code Sign

KMES Host:

Expected authority at index 2: //

Issuance Policy:

Hash Algorithm: MD5

Poll Interval (in seconds):

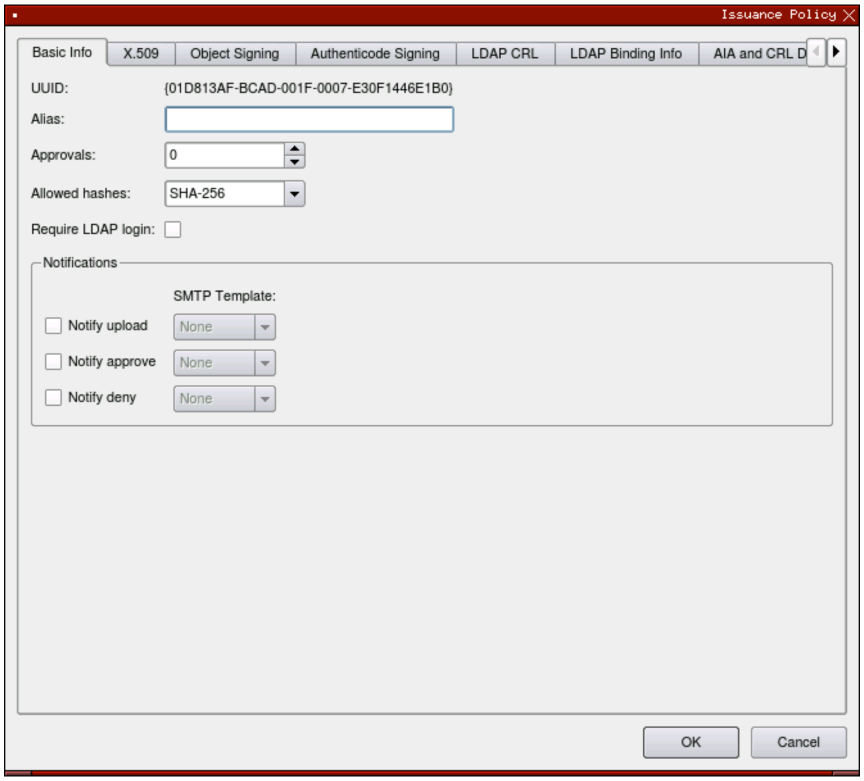
Field is required

TLS PKI: - none -

Credentials: - none -

Files to sign

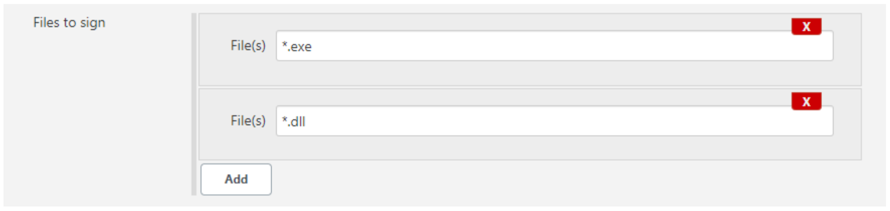
4. In the **Method of Signature** field, leave the default value (i.e., **Code Sign**).
   NOTE: There are currently two types of signatures: **Code Sign** and **External Signature**. Code Sign will try to use knowledge of the file format to embed a signature. If it does not understand the file format, it will fail. An external signature does not need to know the file format, but it cannot embed signatures.
5. In the **KMES Host** field, enter the KMES host to connect to. The port number is optional. It will default to port 2001, the System/Host API port, which is the port that we want to connect to.
6. In the **Issuance Policy** field, enter the UUID of the issuance policy to handle the signing request. To get this information, log in to the KMES application interface, then navigate to the *Certificate Authorities* menu. Right-click on the **Root** Jenkins code signing certificate that is under the **Jenkins** Certificate Container, then select *Issuance Policy -> Edit*. Note down the UUID that is in the first field of the *Basic Info* tab, as seen below:



UUID: {01D813AF-BCAD-001F-0007-E30F1446E1B0}
Alias:
Approvals: 0
Allowed hashes: SHA-256
Require LDAP login: ☐
Notify upload ☐ None
Notify approve ☐ None
Notify deny ☐ None

Back in the Jenkins GUI, enter the UUID in the Issuance Policy field.

7. In the **Hash Algorithm** field, select the hash algorithm to use when requesting signatures.
   NOTE: The hash algorithm that you select must be one of the allowed hashes that you configured for the Issuance Policy attached to the **Root** Jenkins code signing certificate under the **Jenkins** Certificate Container.
8. In the **Poll Interval** field, specify the amount of time in seconds that you want the FXCL Jenkins plugin to wait between code signing status requests that it sends to the KMES.
9. In the **TLS PKI** field, click the dropdown and select the TLS PKI that was imported as a PKCS #12 file in a previous section.
10. In the **Credentials** field, select the username with password credentials configured in section 4.4.
11. In the **Files to sign** field, click the **Add** button. Then, in the **File(s)** field, enter "*.exe".
    NOTE: Multiple files can be added, and the asterisk (*) regular expression is supported as well. For example, you could configure it as shown below if you want all .exe and .dll files in the project to be signed.



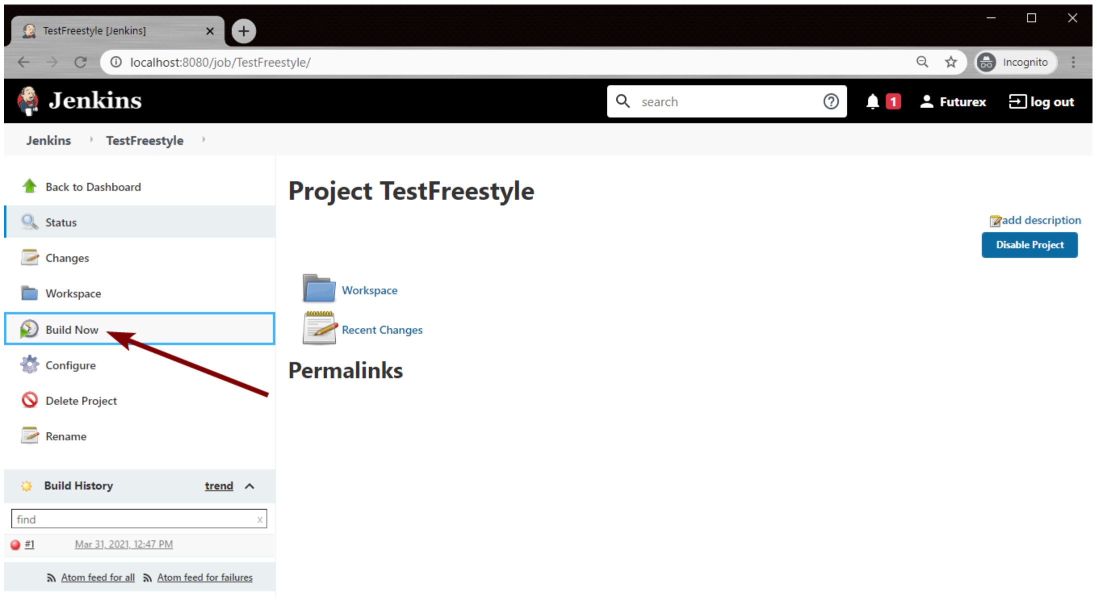
File(s) *.exe
File(s) *.dll

12. Click the **Save** button at the bottom of the page. This will take you back to the main page for the Freestyle project.

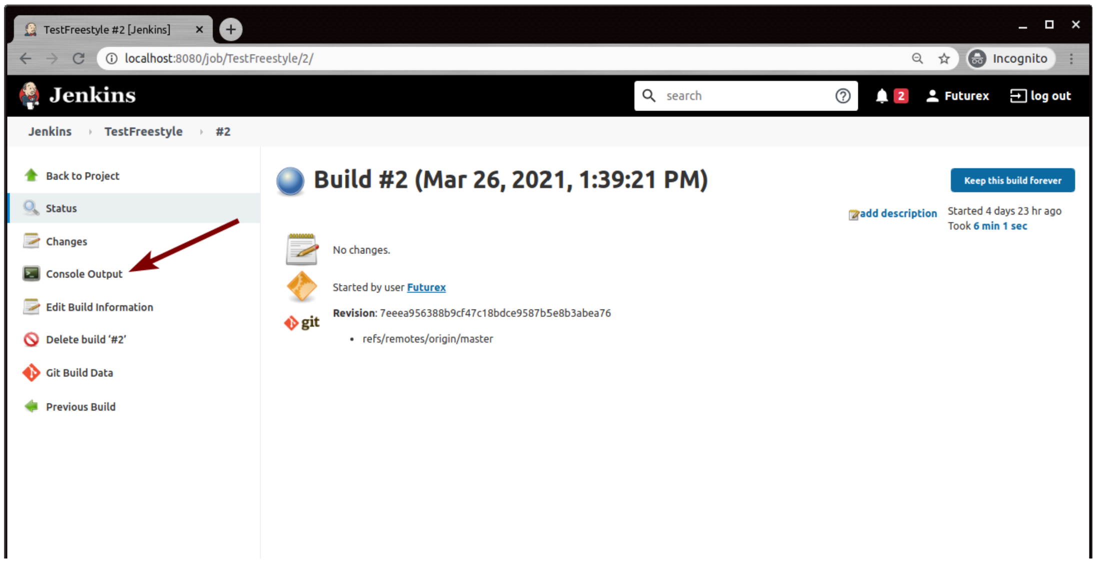
[4.5.2] Testing a KMES code signing by running the Freestyle project

NOTE: Before proceeding with the steps in this section, copy any **.exe** file to the root directory of the Freestyle project (it can be any legitimate .exe file). If you do not complete this step, the build will fail because the KMES will not have any files to sign.

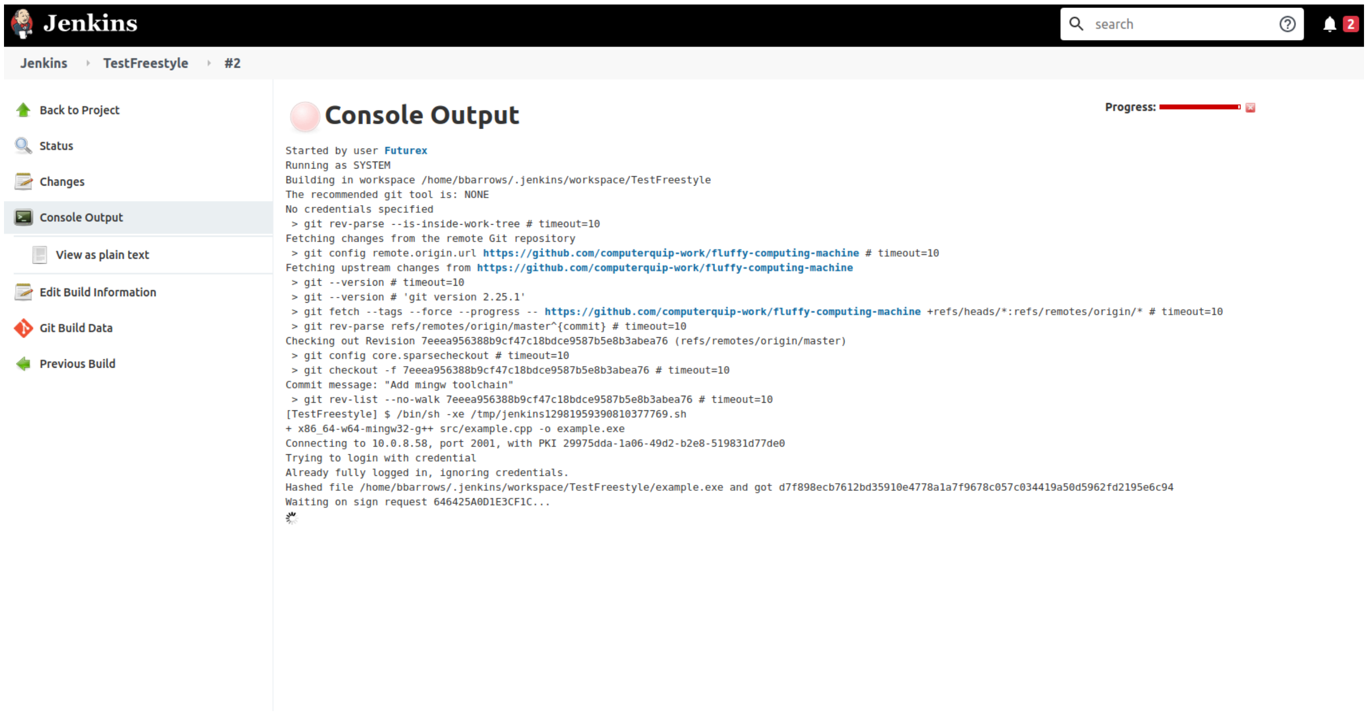
1. From the Freestyle project's main page, click **Build Now** in the left-hand menu.



2. From the main page for the build that was just initiated, click **Console Output** in the left-hand menu.



3. You should see something similar to the following in the console output:

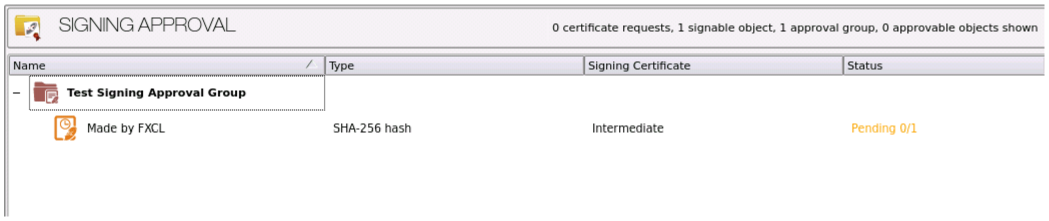


Console Output

```
Started by user Futurex
Running as SYSTEM
Building in workspace /home/bbarrows/.jenkins/workspace/TestFreestyle
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/computerquip-work/fluffy-computing-machine # timeout=10
Fetching upstream changes from https://github.com/computerquip-work/fluffy-computing-machine
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/computerquip-work/fluffy-computing-machine +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 7eeea956388b9cf47c18bdce9587b5e8b3abea76 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 7eeea956388b9cf47c18bdce9587b5e8b3abea76 # timeout=10
Commit message: "Add mingw toolchain"
 > git rev-list --no-walk 7eeea956388b9cf47c18bdce9587b5e8b3abea76 # timeout=10
[TestFreestyle] $ /bin/sh -xe /tmp/jenkins12981959390810377769.sh
+ x86_64-w64-mingw32-g++ src/example.cpp -o example.exe
Connecting to 10.0.8.58, port 2001, with PKI 29975dda-1a06-49d2-b2e8-519831d77de0
Trying to login with credential
Already fully logged in, ignoring credentials.
Hashed file /home/bbarrows/.jenkins/workspace/TestFreestyle/example.exe and got d7f898ecb7612bd35910e4778a1a7f9678c057c034419a50d5962fd2195e6c94
Waiting on sign request 646425A0D1E3CF1C...
```

The last line in the output says, "Waiting on sign request 646425A0D1E3CF1C". This means that there were no errors on the Jenkins side, and the signing request was submitted successfully.

4. We need to log back in to the KMES now to approve the signing request. Once logged in, navigate to the Signing Approval menu. There you should see something similar to the following:

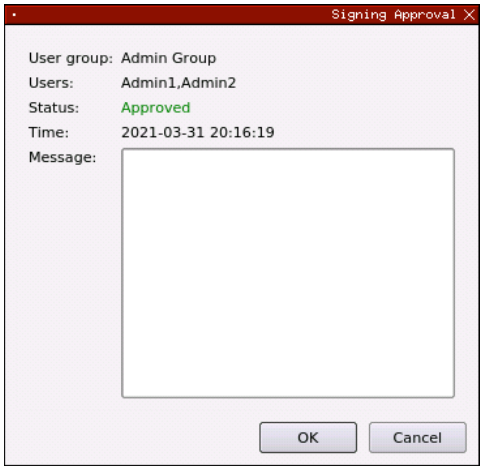


0 certificate requests, 1 signable object, 1 approval group, 0 approvable objects shown

Name	Type	Signing Certificate	Status
Test Signing Approval Group			
Made by FXCL	SHA-256 hash	Intermediate	Pending 0/1

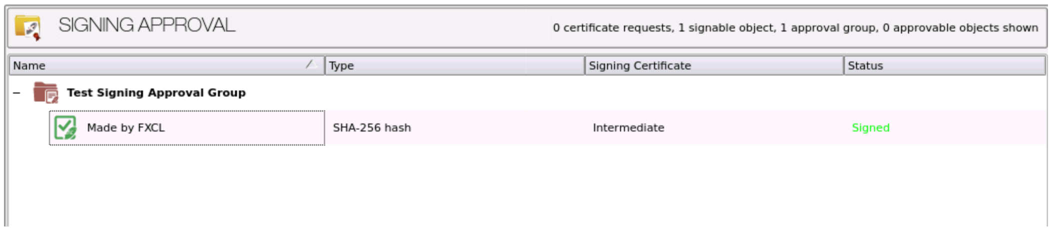
5. Right-click on the signable object that's under the Approval Group you created, then select **Approve...**

6. A box should pop up, showing that the signing request was approved:



User group: Admin Group
Users: Admin1,Admin2
Status: Approved
Time: 2021-03-31 20:16:19
Message:

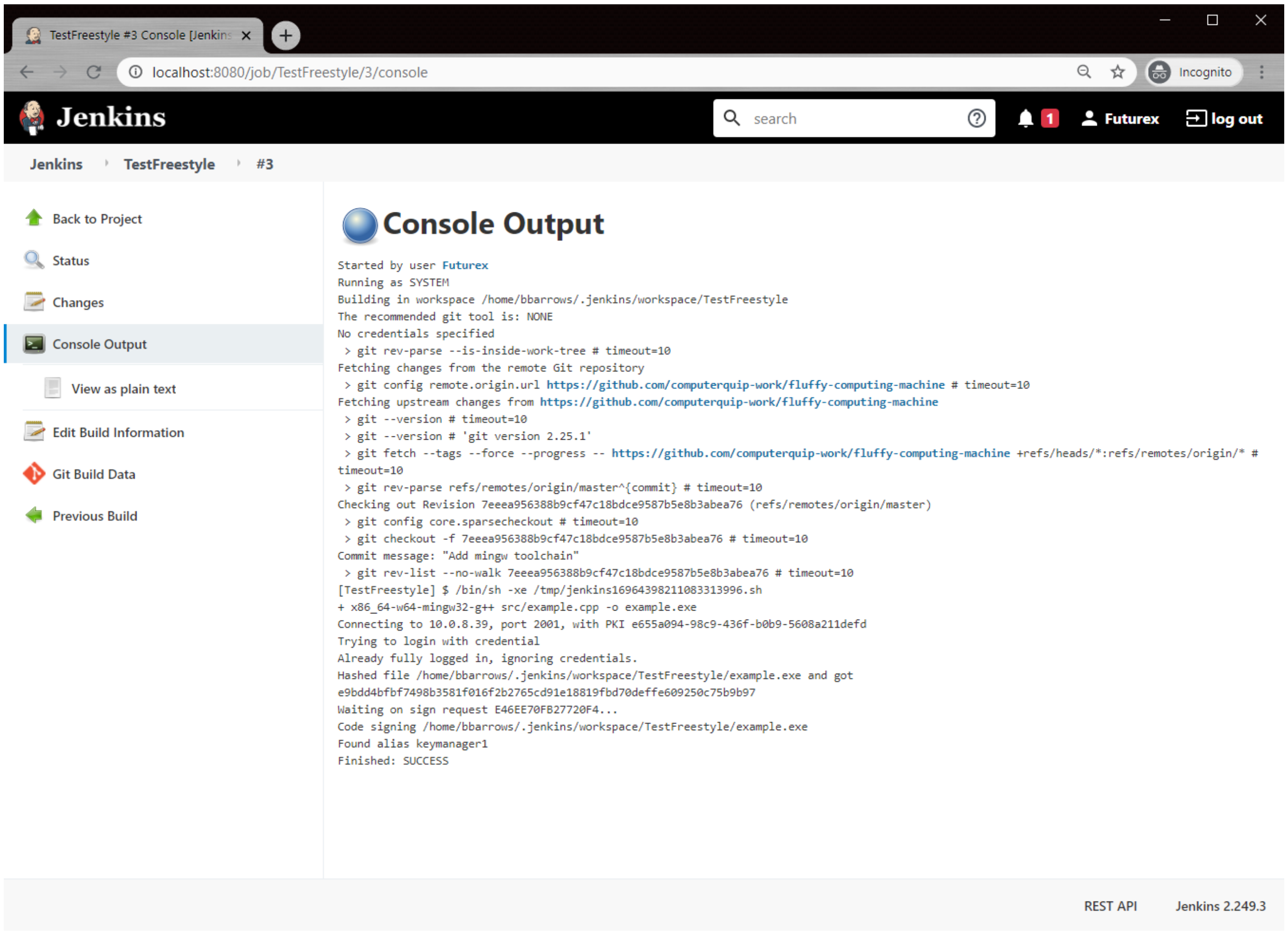
7. Click OK and you'll see that the signing request now has a green checkmark beside it.



SIGNING APPROVAL — 0 certificate requests, 1 signable object, 1 approval group, 0 approvable objects shown

Name	Type	Signing Certificate	Status
Test Signing Approval Group			
Made by FXCL	SHA-256 hash	Intermediate	Signed

8. Return to the Jenkins GUI. After the FXCL Jenkins Plugin has polled the KMES again for the status of the signing request, it should complete the code signing process and finish with a "SUCCESS" message, as shown below:



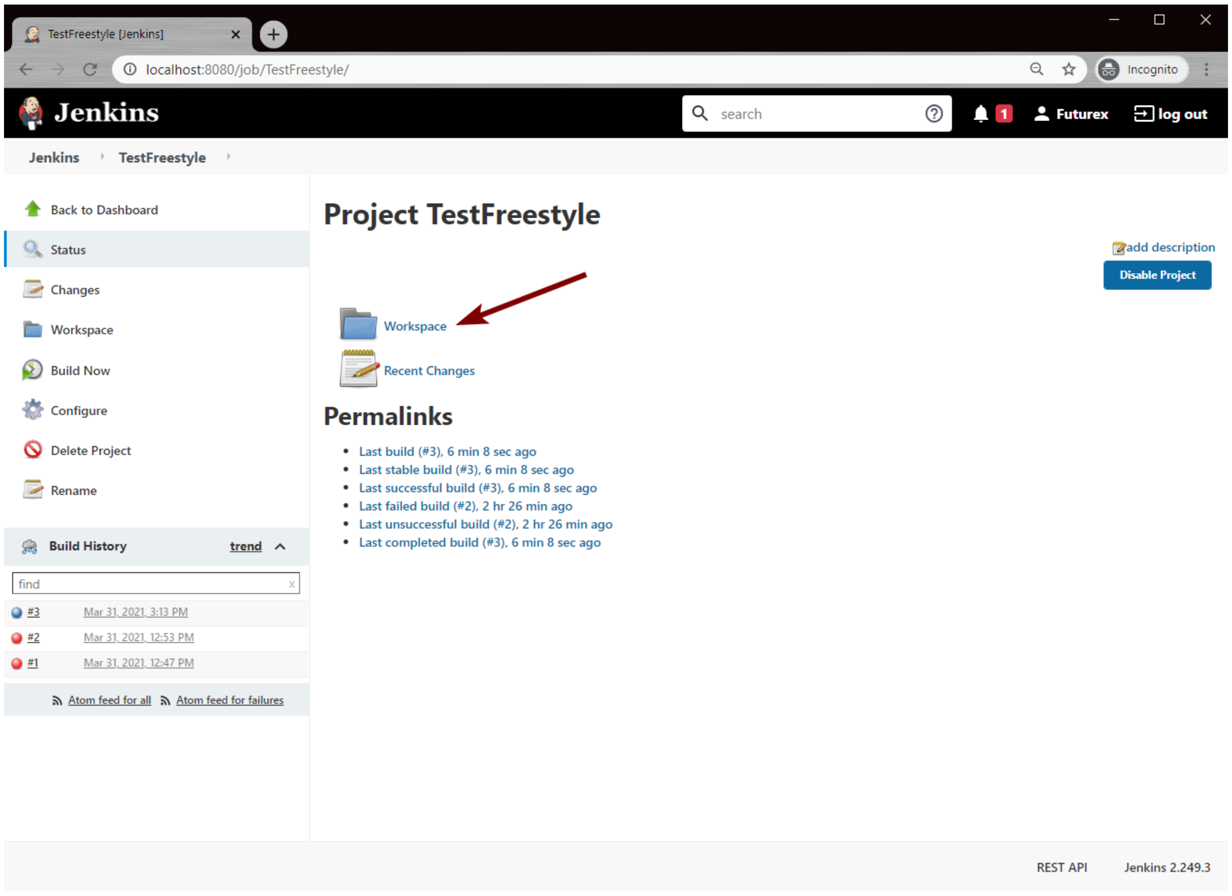
Console Output

```
Started by user Futurex
Running as SYSTEM
Building in workspace /home/bbarrows/.jenkins/workspace/TestFreestyle
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/computerquip-work/fluffy-computing-machine # timeout=10
Fetching upstream changes from https://github.com/computerquip-work/fluffy-computing-machine
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/computerquip-work/fluffy-computing-machine +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 7eeea956388b9cf47c18bdce9587b5e8b3abea76 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 7eeea956388b9cf47c18bdce9587b5e8b3abea76 # timeout=10
Commit message: "Add mingw toolchain"
 > git rev-list --no-walk 7eeea956388b9cf47c18bdce9587b5e8b3abea76 # timeout=10
[TestFreestyle] $ /bin/sh -xe /tmp/jenkins16964398211083313996.sh
+ x86_64-w64-mingw32-g++ src/example.cpp -o example.exe
Connecting to 10.0.8.39, port 2001, with PKI e655a094-98c9-436f-b0b9-5608a211defd
Trying to login with credential
Already fully logged in, ignoring credentials.
Hashed file /home/bbarrows/.jenkins/workspace/TestFreestyle/example.exe and got
e9bdd4bfbf7498b3581f016f2b2765cd91e18819fbd70deffe609250c75b9b97
Waiting on sign request E46EE70FB27720F4...
Code signing /home/bbarrows/.jenkins/workspace/TestFreestyle/example.exe
Found alias keymanager1
Finished: SUCCESS
```

[4.5.3] Confirming that the .exe file is signed

NOTE: The following example is in Windows 10. The process for confirming whether a file is signed will vary depending on which operating system you are using.

1. Navigate back to the main page for the Freestyle project, then click on the **Workspace** folder.



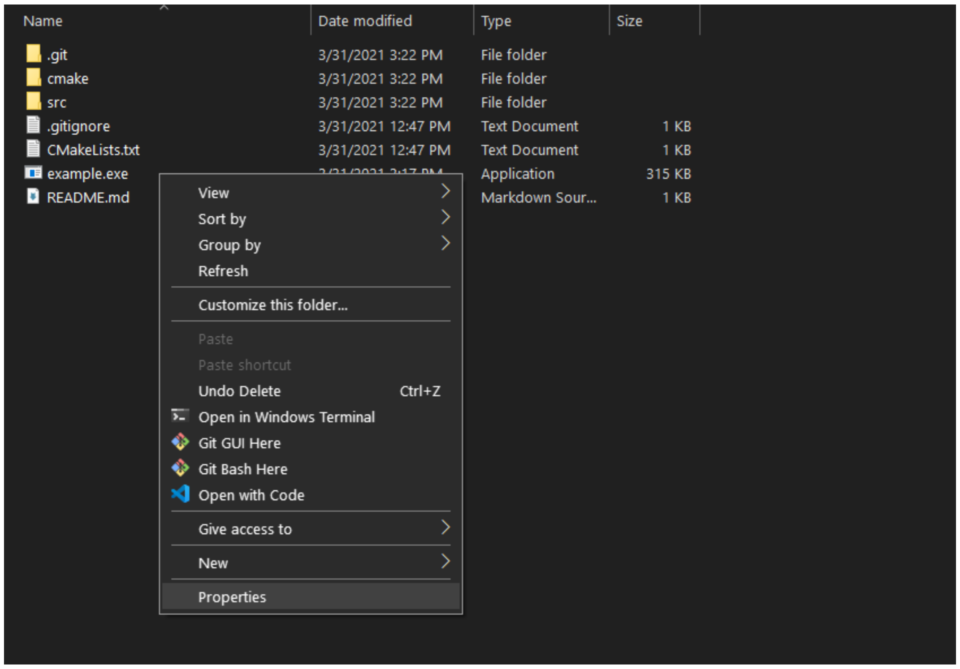
Project TestFreestyle

Permalinks

- Last build (#3), 6 min 8 sec ago
- Last stable build (#3), 6 min 8 sec ago
- Last successful build (#3), 6 min 8 sec ago
- Last failed build (#2), 2 hr 26 min ago
- Last unsuccessful build (#2), 2 hr 26 min ago
- Last completed build (#3), 6 min 8 sec ago

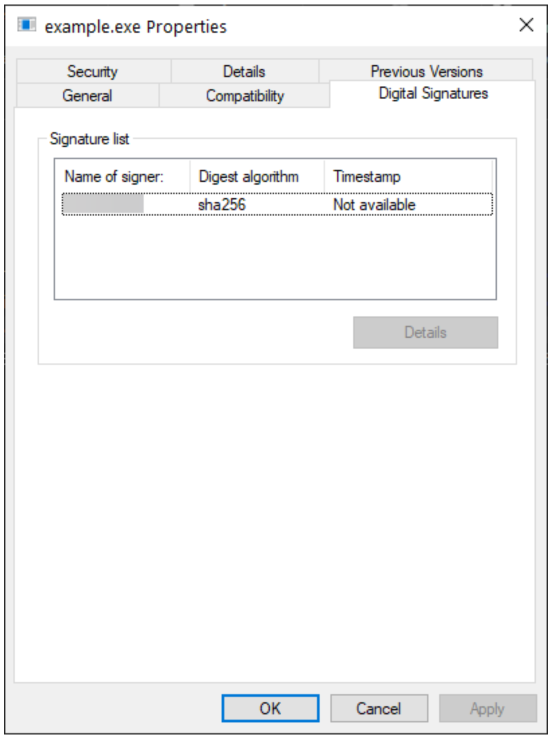
#3	Mar 31, 2021, 3:13 PM
#2	Mar 31, 2021, 12:53 PM
#1	Mar 31, 2021, 12:47 PM

2. Click the **(all files in zip)** button in the center of the page to download a zip of all files in the workspace.
3. In your file manager, navigate to where you downloaded the zip file to, then extract it.
4. Navigate into the folder that was extracted, right-click on the .exe file that was signed, and select **Properties**.



Name	Date modified	Type	Size
.git	3/31/2021 3:22 PM	File folder	
cmake	3/31/2021 3:22 PM	File folder	
src	3/31/2021 3:22 PM	File folder	
.gitignore	3/31/2021 12:47 PM	Text Document	1 KB
CMakeLists.txt	3/31/2021 12:47 PM	Text Document	1 KB
example.exe	3/31/2021 3:17 PM	Application	315 KB
README.md		Markdown Sour...	1 KB

5. In the Properties dialog, navigate to the **Digital Signatures** tab. There you can see the name of the certificate that signed the file. To retrieve more details you can select the signature, then click the **Details** button and you will be able to view information such as the validity dates of the certificate that signed the file, the signature hash algorithm that was used, etc.



Name of signer:	Digest algorithm	Timestamp
[redacted]	sha256	Not available

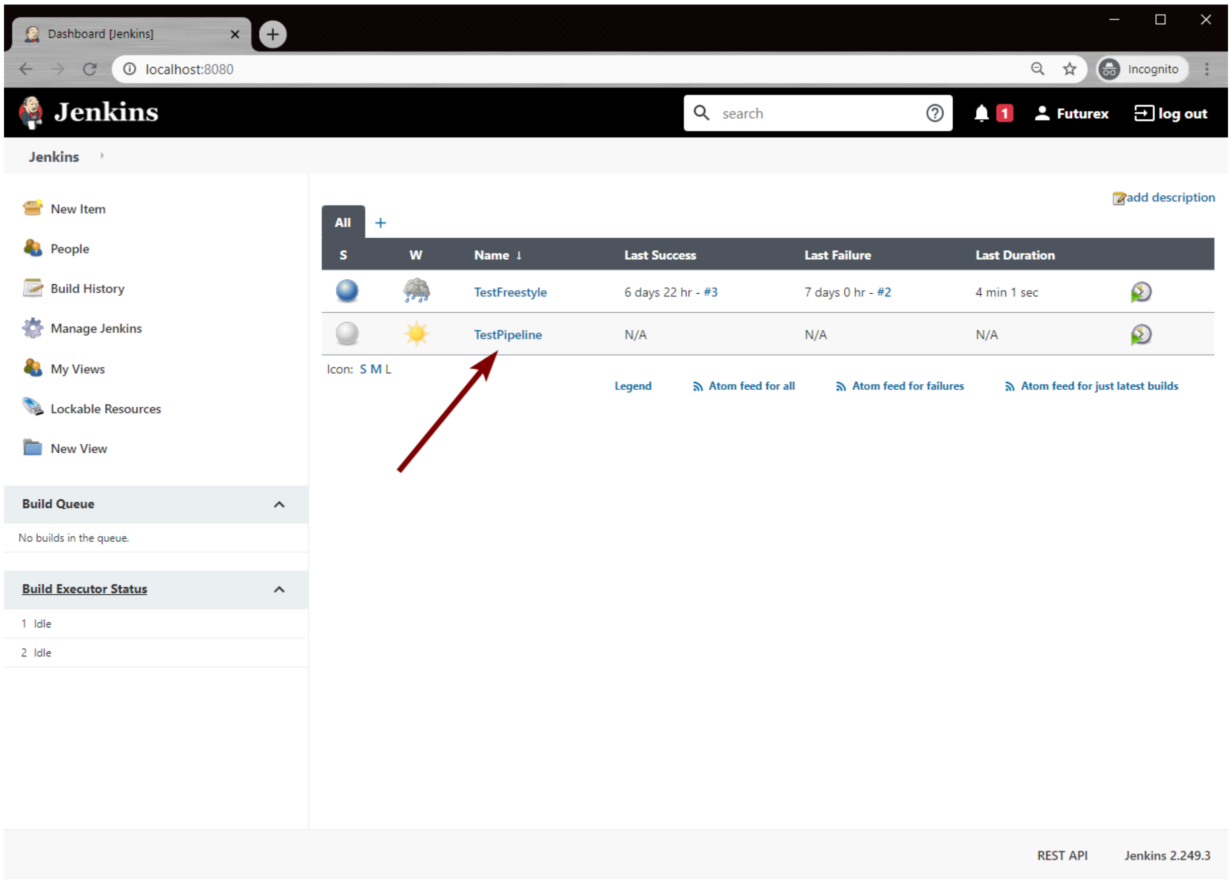
[4.6] USING THE FXCL JENKINS PLUGIN SYNTAX GENERATOR

There is another type of project in Jenkins calls a "Pipeline" project. Essentially, it is a scriptable version of a project. Jenkins describes a Pipeline project this way: "Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type."

Another feature of the FXCL Jenkins Plugin is a syntax generator. It is intended to be used within the context of Pipeline projects. It makes it simple and easy to generate a script for automating code signing, which can be added to existing Pipeline scripting code.

The following steps walk through a basic tutorial of how to use the FXCL Jenkins Plugin syntax generator:

1. From the main Jenkins dashboard, click on an existing Pipeline project.

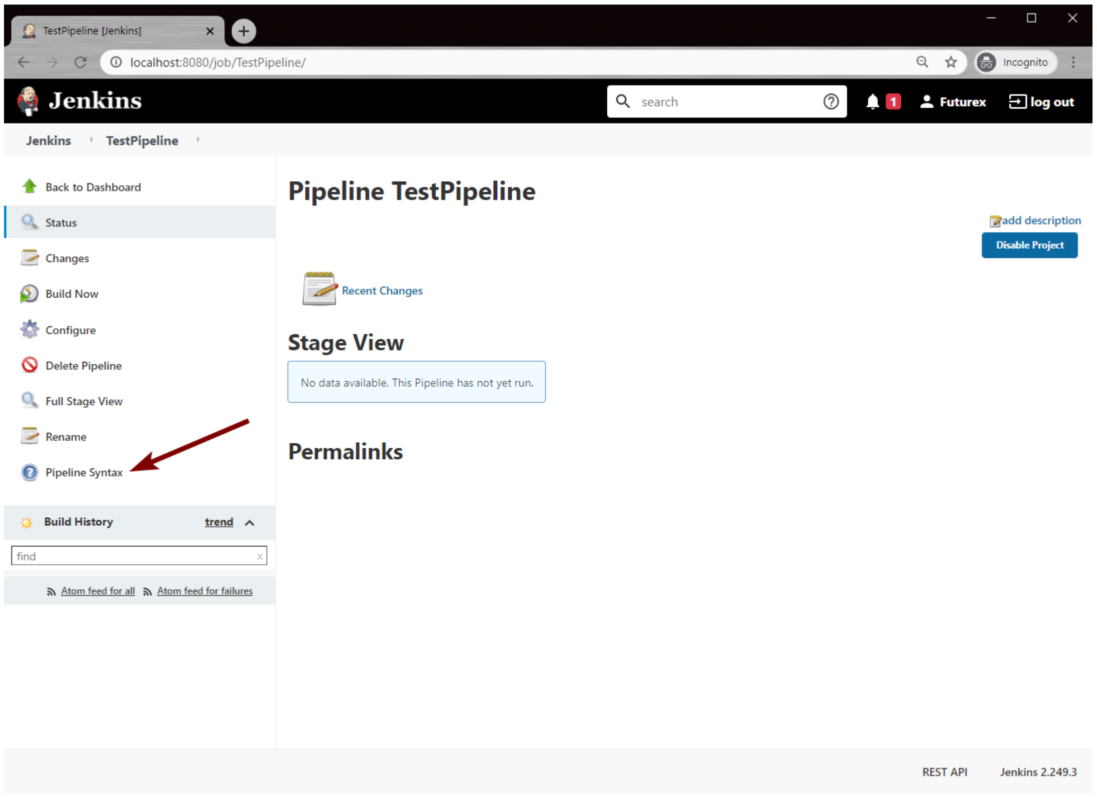


No builds in the queue.

1 Idle
2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
		TestFreestyle	6 days 22 hr - #3	7 days 0 hr - #2	4 min 1 sec
		TestPipeline	N/A	N/A	N/A

2. On the Pipeline project's main page, click on **Pipeline Syntax** in the left-hand menu.



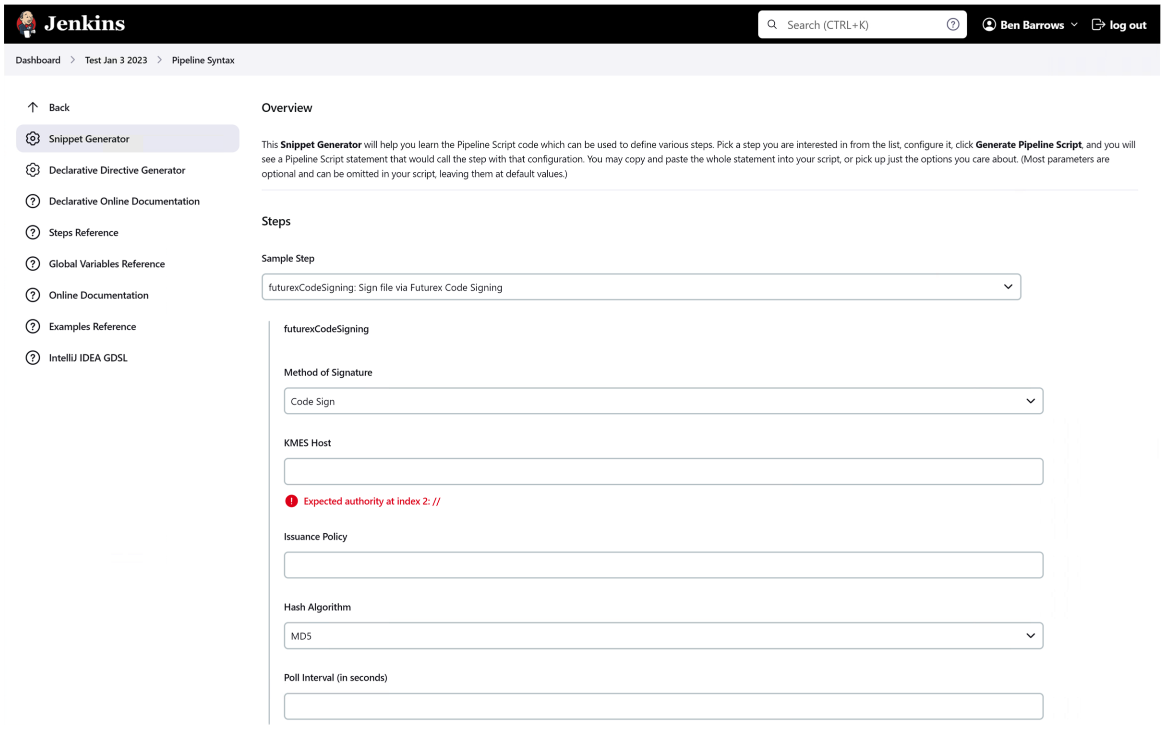
Pipeline TestPipeline

Stage View

No data available. This Pipeline has not yet run.

Permalinks

3. In the Steps section, click on the **Sample Step** dropdown and select **kmesCodeSign: Sign file via registration authority**.


Jenkins
Search (CTRL+K)
Ben Barrows
log out
Dashboard › Test Jan 3 2023 › Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: futurexCodeSigning: Sign file via Futurex Code Signing

futurexCodeSigning

Method of Signature: Code Sign

KMES Host:

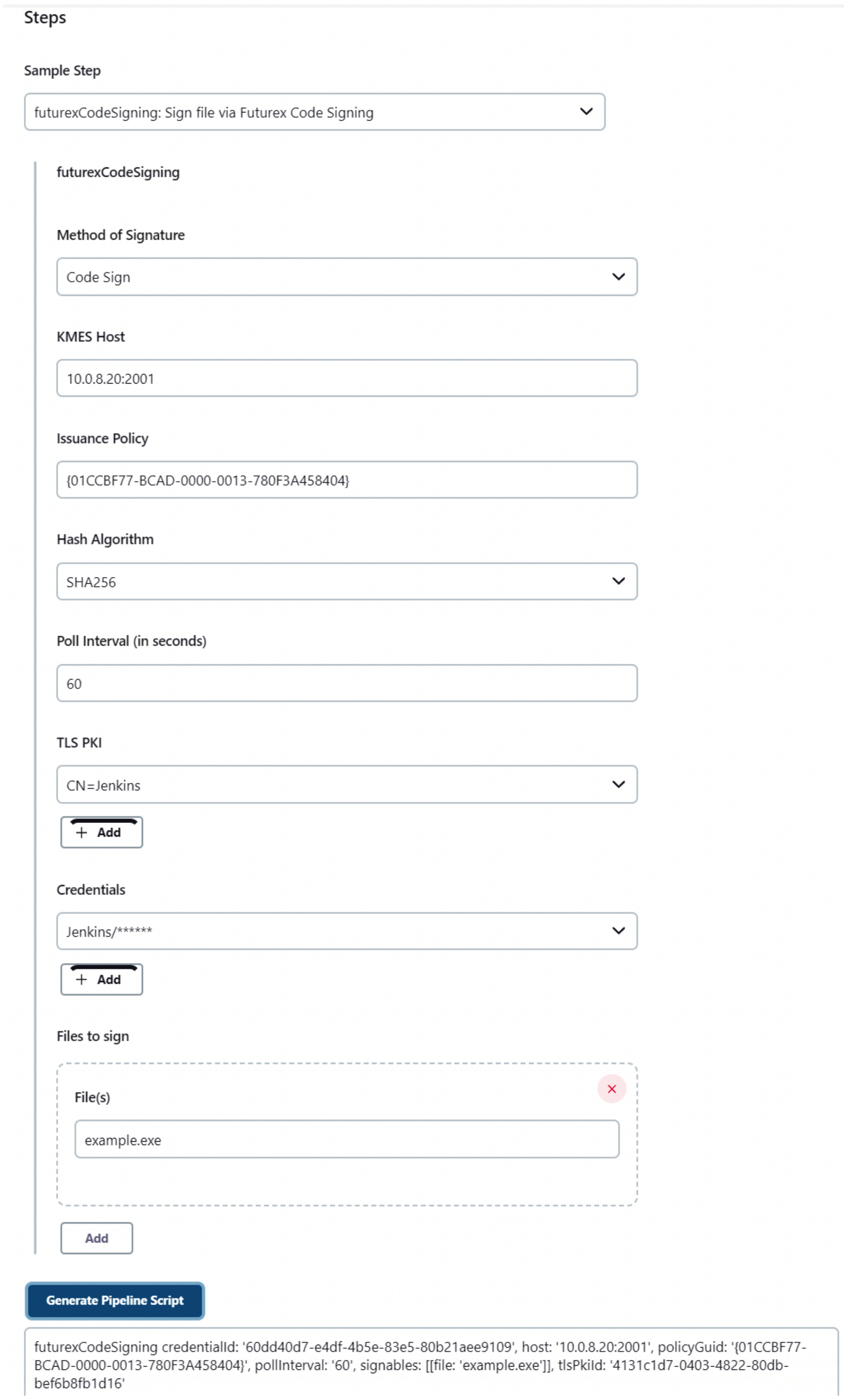
Expected authority at index 2: //

Issuance Policy:

Hash Algorithm: MD5

Poll Interval (in seconds):

4. The fields that need to be filled in our identical to the fields that were filled in for signing files via registration authority in the Freestyle project example. Once you've filled in every field, click the **Generate Pipeline Script** button. This will generate the syntax needed to script code signing within your Pipeline project, as shown below:


Steps

Sample Step: futurexCodeSigning: Sign file via Futurex Code Signing

futurexCodeSigning

Method of Signature: Code Sign

KMES Host: 10.0.8.20:2001

Issuance Policy: {01CCBF77-BCAD-0000-0013-780F3A458404}

Hash Algorithm: SHA256

Poll Interval (in seconds): 60

TLS PKI: CN=Jenkins

Credentials: Jenkins/******

Files to sign

File(s): example.exe

```
futurexCodeSigning credentialId: '60dd40d7-e4df-4b5e-83e5-80b21aee9109', host: '10.0.8.20:2001', policyGuid: '{01CCBF77-BCAD-0000-0013-780F3A458404}', pollInterval: '60', signables: [[file: 'example.exe']], tlsPkiId: '4131c1d7-0403-4822-80db-bef6b8fb1d16'
```

5. Then, simply copy and paste the syntax that was generated into an existing Pipeline script to automate code signing within your project.

APPENDIX A: XCEPTIONAL SUPPORT

In today's high-paced environment, we know you are looking for timely and effective resolutions for your mission-critical needs. That is why our Xceptional Support Team does whatever it takes to ensure you have the best experience and support possible. Every time. Guaranteed.

- 24x7x365 mission critical support
- Level 1 to level 3 support
- Extremely knowledgeable subject matter experts

At Futurex, we strive to supply you with the latest data encryption innovations as well as our best-in-class support services. Our Xceptional Support Team goes above and beyond to meet your needs and provide you with exclusive services that you cannot find anywhere else in the industry.

- Technical Services
- Onsite Training
- Virtual Training
- Customized Consulting
- Customized Software Solutions
- Secure Key Generation, Printing, and Mailing
- Remote Key Injection
- Certificate Authority Services

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com