# Sling IDE tooling - Signing



## Overview

All Eclipse plugins should be signed with the means outlined at https://docs.oracle.com/javase/tutorial/deployment/jar/intro.html in order to not emit warnings during installation. This signing is orthogonal to the PGP signing of ASF release artifacts (which is for source artifacts only)

The standard tool for signing Eclipse plugins is https://docs.oracle.com/javase/8/docs/technotes/tools/windows/jarsigner.html which emits either

1. Digital Signature Algorithm (DSA) with the SHA1 digest algorithm
2. RSA algorithm with the SHA256 digest algorithm
3. Elliptic Curve (EC) cryptography algorithm with the SHA256 with Elliptic Curve Digital Signature Algorithm (ECDSA).

## ASF Signature Service

ASF leverages Digicert One as code signing service (https://infra.apache.org/digicert-access.html).

Sling has been assigned a dedicated key pair by INFRA in **INFRA-23724** - Getting issue details... STATUS which is accessible at https://one.digicert.com/signingmanager/keypairs/be8507fa-dd5c-436c-b0d8-c43708d18a49?alias=Sling-PMC-2022-09. This keypair is a ***3072 bit RSA*** key, therefore the signature algorithm 2 outlined above is used when signing JARs.

Instead of manually downloading the certificate the code signing uses a custom PKCS#11 implementation which leverages a ReST service. That allows to keep the private key with DigiCert only and not requires anyone to download it to a local computer (or the ASF Jenkins).

**ASF pays per signature therefore only releases should be signed (but not CI build artifacts)**

## Maven Plugin

The recommended approach from Digicert is leveraging the custom PKCS#11 implementation with the `maven-jarsigner-plugin` for automating the JAR signing process: https://docs.digicert.com/en/software-trust-manager/signing-tools/sign-java-files-with-jarsigner-using-pkcs11-integration.html. (ASF IN FRA also has some recommendations at https://infra.apache.org/digicert-use.html, but those are primarily targeted at signing Windows applications at the moment).

The approach from DigiCert requires installation of the custom PKCS#11 library on at least one Jenkins node which does the signing (requested in **INFRA-23844** - Getting issue details... STATUS ).

In addition some environment variables need to be set populated from Jenkins credentials maintained on organization level at https://ci-builds.apache.org/job/Sling/credentials/. The conversion from credentials to environment variable is being done by https://www.jenkins.io/doc/pipeline/steps/credentials-binding/.

All secrets were created by INFRA in the context of **INFRA-23844** - Getting issue details... STATUS

These are

| | Environment Variable | Description | Jenkins Credentials ID |
|---|---|---|---|
| 1 | SM_API_KEY | is the API token generated as outlined at https://docs.digicert.com/de/digicert-one/secure-software-manager/ci-cd-integrations/maven-integration-with-pkcs11.html.<br>The Token is bound to a service user created by INFRA (https://one.digicert.com/account/access/service-user/0f81f60f-ad92-469e-8db0-4ed91f9b0f55) | sling-digicert-pkcs-api-key (https://ci-builds.apache.org/job/Sling/credentials/store/folder/domain/_/credential/sling-digicert-pkcs-api-key/) |
| 2 | SM_CLIENT_CERT_FILE | The path to the certificate to use for the client authentication<br><br>The certificate is bound to a service user created by INFRA (https://one.digicert.com/account/access/service-user/0f81f60f-ad92-469e-8db0-4ed91f9b0f55) | sling-digicert-pkcs-certificate (https://ci-builds.apache.org/job/Sling/credentials/store/folder/domain/_/credential/sling-digicert-pkcs-certificate/) |

| | | | |
|---|---|---|---|
| 3 | SM_CLIENT_CERT_PASSWORD | The password of the certificate to use for the client authentication | sling-digicert-pkcs-cert-pw(https://ci-builds.apache.org/job/Sling/credentials/store/folder/domain/_/credential/sling-digicert-pkcs-cert-pw/) |
| 4 | SM_HOST | https://clientauth.one.digicert.com (has to be set explicitly as the default host does not allow authentication with SM_CLIENT_CERT_FILE) | n/a |

# Integration in Tycho P2 Repository Build

Instead of signing each plugin individually, this is done centrally while building the P2 repository. Only that module is built separately on the Jenkins node having the PKCS#11 library (label=pkcs11)

Further infos at https://github.com/eclipse-tycho/tycho/discussions/1685.

# Known Issues

- The provided PKCS#11 library from https://one.digicert.com/signingmanager/client-tools/smpkcs11-mac-x64 is not suitable for usage on Apple Silicon (aarch64). jarsigner then emits:
  `jarsigner error: java.security.ProviderException: Initialization failed`
  - Version 1.32.0 fixes this as it comes with a dylib for both architectures X86-64 and ARM64
- The provided PKCS#11 library crashes with a segfault on X86-64 executed with Rosetta 2 on Apple Silicon when calling jarsigner.
  - This is only an issue with Java 19 or newer. Older versions should work fine.

# Migration to GPG Signatures

P2 supports also GPG signatures: https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fp2_pgp.html

Migrating to it would streamline and consolidate the signing process between ASF/Maven Central and P2.

It has the drawback, that there is no built-in trust and that the keys used for signing are not directly affiliated with ASF.

This is tracked in **SLING-11680** - Getting issue details... STATUS