
# Integración Gradle - Jenkins - Sonar

## Oficina de Calidad

## Versión: 2.3.0

# Control de versiones

| Versión | Fecha | Autor | Descripción / Comentarios |
|---|---|---|---|
| 1.00 | 01/02/2017 | SGTIC | Primera versión |
| V2_v01 | 05/08/2017 | Oficina de Calidad | Se ajusta a la Normativa de configuración Gradle |
| V2_v02 | 04/09/2020 | Oficina de Calidad | Se cambian de nombre todas las referencias al documento MIR-INT-NORMATIVA-GRADLE.docx por MIR-INT-NORMA-GRADLE.docx |
| 2.3.0 | 19/02/2021 | Oficina de Calidad | Modificación del plugin sonarqube |

# Índice

# 1. INTRODUCCIÓN

## 1.1. OBJETO

El objeto del presente documento es la definición de requisitos para la integración de proyectos de desarrollo con las herramientas corporativas de automatización de tareas (Gradle), plataforma de integración continua (Jenkins) y evaluación continua de la calidad del código (SonarQube).

## 1.2. ALCANCE

Este documento afecta a las áreas, servicios y personas involucradas en el desarrollo de aplicaciones para la SGSICS del MIR.

## 1.3. GLOSARIO

| Término | Descripción |
|---|---|
| SGSICS | Subdirección General De Sistemas De Información Y Comunicaciones Para La Seguridad. |
| MIR | Ministerio del Interior |

## 1.4. DOCUMENTOS RELACIONADOS

| /Ubicación/Documento | Descripción |
|---|---|
| PublicacionIntranet/plantillas/doc/03.CSI/MIR-APLIC-CSI-DOC_COMPILACION.docx | Documento que describe los pasos a seguir para realizar la compilación del sistema de información. |
| PublicacionIntranet/03.CSI/MIR-INT-NORMA-GRADLE.docx | Documento que describe la normativa a seguir por todos los proyectos configurados con Gradle. |
| PublicacionIntranet/20-ANEXOS/MIR-INT-CONFIGURACION-SONARLINT.docx | Documento que sirve de guía para configurar SonarLint en todos los IDE (Eclipse en el caso de la SGSICS) utilizados para los proyectos Java desarrollados. |

# 2. GRADLE

## 2.1. CONFIGURAR GRADLE Y JAVA EN LOCAL

### 2.1.1. Configurar entorno Local Gradle

La oficina de Calidad, a través de la publicación de la metodología del MIR, facilitará al equipo de desarrollo el fichero de configuración de Gradle (*gradle.properties*) con los parámetros necesarios para el acceso a los diferentes servicios (Artifactory) así como el parámetro requerido para la localización de la Java Developer Kit (jdk) para la compilación de los fuentes (javac_jdk6, javac_jdk7, javac_jdk8,..)

```
##gradle.properties##
…
artifactoryUser=## USUARIO_DOMINIO_MIR ##
artifactoryPassword=## PASSWORD_DOMINIO_MIR ##
artifactoryUrl=https://sqa-servicio.mir.es/artifactory
…
pathJdk6=## JAVA_HOME ##
pathJdk7=## JAVA_HOME ##
pathJdk8=## JAVA_HOME ##
```

Este fichero deberá ser personalizado localmente por cada desarrollador con sus propias credenciales para el dominio MIR. Por dicho motivo es conveniente ubicar el fichero en el directorio home de gradle del usuario (gradle user home) en lugar de incluir el mismo en el directorio de construcción del proyecto (project build dir)

En el siguiente ejemplo se muestra el uso de los parámetros de configuración dentro del fichero *build.gradle*

```
buildscript {
    repositories {
        maven {
            url "${artifactoryUrl}"
            credentials {
                username "${artifactoryUser}"
                password "${artifactoryPassword}"
            }
        }
    }
    dependencies {
```

```
        classpath "net.saliman:gradle-properties-plugin:1.4.5"
    }
}
……

compileJava{

    options.encoding = projectInfo.encoding
    options.fork=true
    options.forkOptions.executable = projectBuildInfo.pathJavac
}
…..
allprojects {
    repositories {
        maven {
            url "${artifactoryUrl}/plugins-release"
            credentials {
                username "${artifactoryUser}"
                password "${artifactoryPassword}"
            }
        }
       maven {
            url "${artifactoryUrl}/libs-release"
            credentials {
                username "${artifactoryUser}"
                password "${artifactoryPassword}"
            }
        }
        maven {
            url "${artifactoryUrl}/libs-snapshot"
            credentials {
                username "${artifactoryUser}"
              password "${artifactoryPassword}"
            }
        }
    }
}
```

La propiedad pathJdkX, se utilizará para la creación de la propiedad pathJavac durante la creación del objeto projectBuildInfo en el fichero environmentBuildConfig. Para obtener más información sobre esta configuración ver el documento MIR-INT-NORMA-GRADLE.docx

## 2.2. INTEGRACIÓN JENKINS

Haciendo uso en el *build.gradle* de los parámetros indicados en el punto anterior, la integración con la plataforma Jenkins será transparente.

En caso de requerirse algún parámetro distinto a los indicados para la construcción del proyecto, dicha necesidad deberá ser comunicada al equipo de calidad con objeto de que sea evaluada y, en caso de considerarse necesario, sea incluido el correspondiente parámetro en la configuración de la plataforma Jenkins para la correcta construcción del proyecto mediante Gradle.

## 2.3. INTEGRACIÓN SONARQUBE

Para la integración con SonarQube será necesario incluir en el fichero *build.gradle* el plugin DSL de Sonar

En el siguiente ejemplo se muestra cómo quedaría el plugin añadido dentro del fichero *build.gradle*

```
plugins {
…
  id "org.sonarqube" version "2.5"
…
}
```

Dicha instrucción deber ubicarse al principio del fichero *build.gradle,* tal y como se indica en el documento MIR-INT-NORMA-GRADLE.docx.

Este plugin permitirá la verificación de la calidad del código mediante la invocación de la tarea “sonarqube” en el entorno de Integración Continua, ejecutado mediante Jenkins al encontrar cambios en el repositorio SVN del proyecto.

- build sonarqube

Esta ejecución no deberá ser lanzada en entornos de desarrollo, ya que la revisión del código se tendrá que realizar mediante SonarLint en tiempo de implementación.

## 2.4. INTEGRACIÓN PLUGINS SONARQUBE (ORG.SONARQUBE)

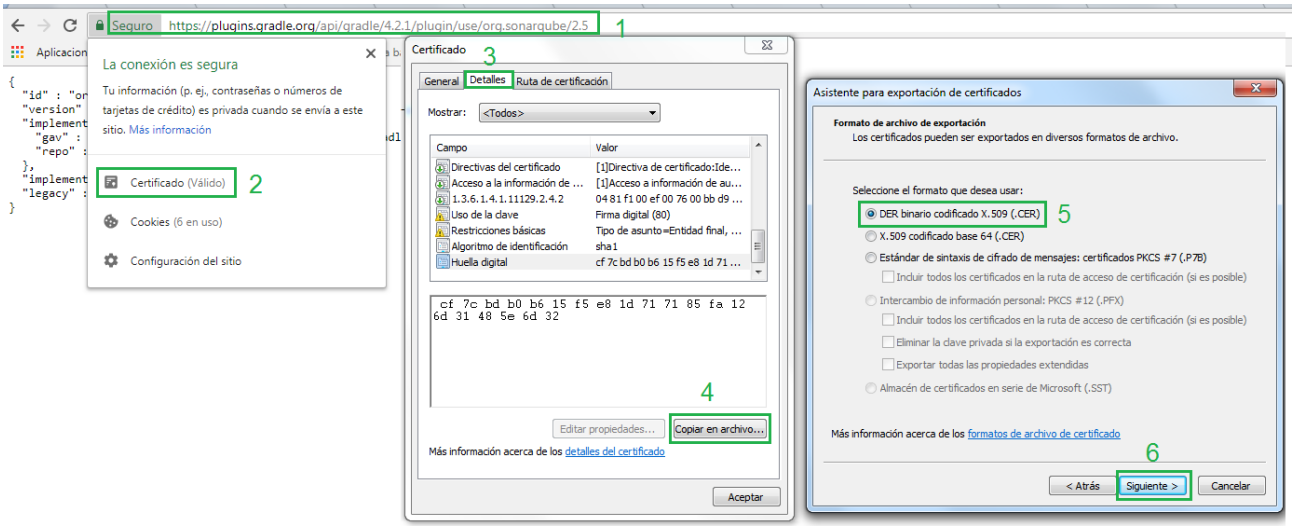
Cuando se realiza una compilación a nivel local (sin usar Jenkins), por ejemplo, “gradle build” puede ocurrir un error que indica que no se puede conectar con el servidor plugins.gradle.org o algún problema con el certificado.

Los pasos a seguir son parecidos al punto “4. Pasos para sincronizar sonarlint con sonarqube”, del documento “MIR-INT-CONFIGURACION-SONARLINT.docx”

Entonces, tendremos que descargar las claves públicas del servidor de plugins de Gradle “plugins.gradle.org” a local e instalarlas en el cacerts de la JRE que ejecuta Gradle.

Para ello debemos seguir los siguientes pasos:

1. Comprobar sobre qué instalación corre la versión de Gradle con la que se está ejecutando el proyecto
    a. gradle --version
2. Descargar la clave pública del servidor
    a. Acceder a https://plugins.gradle.org/api/gradle/4.2.1/plugin/use/org.sonarqube/2.6.2
    b. Descargar la clave pública




3. Instalar la clave pública en el cacerts de la JRE que ejecuta Gradle
    a. keytool -import -v -trustcacerts -alias plugins.gradle.org -file {UBICACIÓN CERTIFICADO DESCARGADO}/{NOMBRE CERTIFICADO DESCARGADO}.cer -keystore {UBICACIÓN JDK}/jre/lib/security/cacerts -keypass changeit -storepass changeit
        i. Ejemplo: keytool -import -v -trustcacerts -alias plugins.gradle.org -file C:/certificado-gradle-plugins.cer -keystore C:/Java/jdk1.8.0_65/jre/lib/security/cacerts -keypass changeit -storepass changeit
4. Comprobar que se ha instalado correctamente
    a. keytool -list -keystore {UBICACIÓN JDK}/jre/lib/security/cacerts
        i. Ejemplo: keytool -list -keystore C:/Java/jdk1.8.0_65/jre/lib/security/cacerts> entradas-cecacert.txt (buscar en el fichero el alias “plugins.gradle.org”)

5. Si todo está correcto, borrar la cache de descargas de dependencias de Gradle (Si no se ha cambiado, por defecto se encuentra en la carpeta del usuario, llamándose .gradle, “C:\Users\{USUARIO}\.gradle”. Borrar todo su contenido)
6. Ejecutar de nuevo el proyecto con Gradle

## 2.5. USO DE GRADLE CON EL SERVIDOR ARTIFACTORY

En los proyectos de desarrollo del MIR se utilizará la herramienta Gradle, para la utilización del servidor indicaremos en el fichero build.gradle del proyecto las siguientes sentencias:

**Para los plugins de Gradle:**

```
buildscript {
    repositories {
        maven {
            url "https://sqa-servicio.mir.es/artifactory/plugins-release"
                credentials {
                    username "${artifactory_user}"
                    password "${artifactory_password}"
                }
            }
    }
}
```

**Para los artefactos de las librerías de Gradle:**

```
repositories {
  maven {
    url "https://sqa-servicio.mir.es/artifactory/libs-release"
    credentials {
       username "${artifactory_user}"
       password "${artifactory_password}"
    }
  }
  maven {
    url "https://sqa-servicio.mir.es/artifactory/libs-snapshot"
    credentials {
      username "${artifactory_user}"
      password "${artifactory_password}"
    }
 }
}
```

El `artifactory_user` se configura en la carpeta de nuestro usuario en Windows en el fichero “gradle.properties” y contendrá las siguientes sentencias:

**C:\Users\<usuario>\.gradle\gradle.properties**

```
org.gradle.daemon=true

artifactory_user=<nuestro usuario_windows>
artifactory_password=<nuestra_password_windows>
```

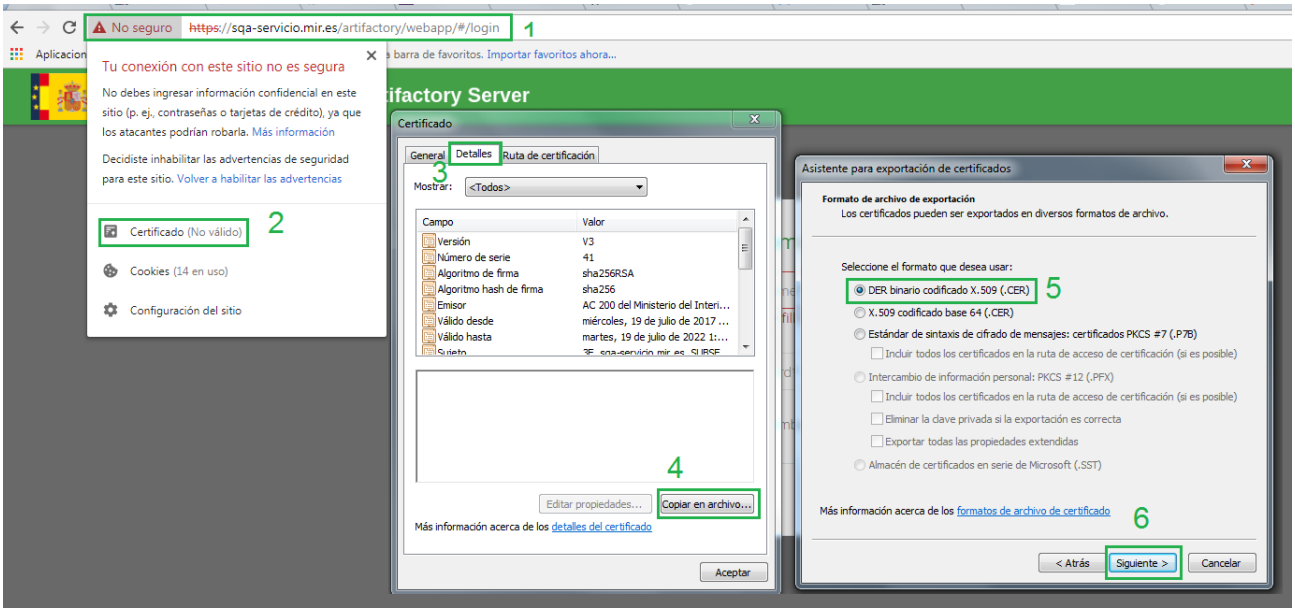
## 2.6. INTEGRACIÓN ARTIFACTORY

Cuando se realiza una compilación a nivel local (sin usar Jenkins), y según los ficheros de configuración de Gradle (build.gradle, gradle.properties, etc), se conectara al servidor Artifactory, y lo mismo que ocurría con el plugins de Sonar (apartado INTEGRACIÓN PLUGINS SONARQUBE (ORG.SONARQUBE)) y sonarlint (documento MIR-INT-CONFIGURACION-SONARLINT.docx) necesita un certificado para realizar la conexión, por lo tanto hay que realizar el mismo procedimiento que sea explicado en los apartados indicados anteriormente.

En este caso, el servidor usado, sqa-servicio.mir.es, es el mismo tanto para Artifactory como para Jenkins, y usan el mismo certificado.

Para ello debemos seguir los siguientes pasos:

1. Descargar la clave pública del servidor
    a. Acceder a https://sqa-servicio.mir.es/artifactory/webapp/#/login
    b. Descargar la clave pública




2. Instalar la clave pública en el cacerts de la JRE que ejecuta Gradle
    a. keytool -import -v -trustcacerts -alias plugins.gradle.org -file {UBICACIÓN CERTIFICADO DESCARGADO}/{NOMBRE CERTIFICADO DESCARGADO}.cer -keystore {UBICACIÓN JDK}/jre/lib/security/cacerts -keypass changeit -storepass changeit
        i. Ejemplo: keytool -import -v -trustcacerts -alias sqa-servicio.mir.es -file C:/certificado-artifactory.cer -keystore C:/Java/jdk1.8.0_65/jre/lib/security/cacerts -keypass changeit -storepass changeit
3. Comprobar que se ha instalado correctamente
    a. keytool -list -keystore {UBICACIÓN JDK}/jre/lib/security/cacerts
        i. Ejemplo: keytool -list -keystore C:/Java/jdk1.8.0_65/jre/lib/security/cacerts> entradas-cecacert.txt (buscar en el fichero el alias “sqa-servicio.mir.es”),

fecha 12/06/2019 la Huella Digital de Certificado (SHA1): 15 fd 56 3b c4 b8 10 41 0c b9 e6 f1 ab 4b 9d 20 2c 9e e2 b1

4. Si todo está correcto, borrar la cache de descargas de dependencias de Gradle (Si no se ha cambiado, por defecto se encuentra en la carpeta del usuario, llamándose .gradle, “C:\Users\{USUARIO}\.gradle”. Borrar todo su contenido)
5. Ejecutar de nuevo el proyecto con Gradle

## 2.7. CHARSET ENCODING

El charset encoding establecido en el MIR para las fuentes y datos es UTF-8. Dicho encoding ha de ser especificado en la tarea de compilación de Gradle para asegurar que todos los ficheros fuente entregados compilan correctamente bajo dicho encoding.

```
compileJava{

    options.encoding = projectInfo.encoding
    options.forkOptions.executable = projectBuildInfo.pathJavac
….
}
```

La propiedad encoding, se deberá configurar en el fichero build.gradle del proyecto principal, asociando el valor “UTF-8” a la propiedad encoding del objeto projectInfo. Para obtener más información de cómo realizar dicha configuración es necesario ver el documento MIR-INT-NORMA-GRADLE.docx