FORTIFY SCA TRANSLATION Java/JSP COOKBOOK SCA V18.20

FORTIFY SCA TRANSLATION Java/JSP COOKBOOK
SCA V18.20

NOTE: This document will not duplicate what is written in the Official documentation.

The official documentation URL:

https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools/1820/

Java Translation Basics

SCA will translate successfully when you have all the Java Source code and do not have to reference any JAR files.
SCA will translate successfully on the command line when you set the CLASSPATH environment variable correctly.
SCA will translate successfully on command line when you set the CLASSPATH with the –cp or –extdirs options correctly
SCA will translate successfully for the following versions:

- Java 5.x
- Java 6x
- Java 7x,
- Java 8x
- Java (including Android) for all of the above versions.

Example:

sourceanalyzer -b myjava8 -source 1.8 -logfile myjava8Translate.log <my java source code directory>

NOTE: The -source option specifies the version of java to translate. This is needed if there are features of a version of java that may have changed from the default 1.8 version. NOTE Java 9 if no longer supported by Oracle.

SCA Java Language Translation Common Errors

There are six common errors with a translation

1. SCA does not support the Java feature or version
2. SCA does not have read access to the files or directory
3. SCA runs out of system resources
4. SCA translate command line option missing or incorrect
5. SCA translate using build tools has some missing references (JARs)
6. SCA has a bug for that function or feature.

SCA Log Files Changes in 18.20

NOTE: See Page 115 in the SCA User Guide for more details.

There are now two log files for each translation starting with version 18.20, a sca.log and a log for fortify support called, sca_FortifySupport.log.

General Translation Tips

Key Translation Onboarding Questions for Java/JSP Applications

The following questions should be asked when setting up a JAVA/JSP application for SCA translation.

1. Where are the Application dependency JAR files located before and during the build process ?
2. Do I have all the source code written by my developers for the application and any third party source that is used in the application that is available ?
3. Do I have all the dependency JAR files locally on my SCA translation system ?
4. How do I ensure that the Java CLASSPATH is set correctly to point to all the required dependency JAR files during the SCA translation process ?

SCA Export Translation For Scan on Another System

The SCA translation dose not require as much resources as the SCA scan process. Exporting a build translation session to another system for scanning that has better processing power with more memory and better processors is a common best practice.

The command line options –make-mobile and –export-build-session are used to move a translation to another machine for scanning.

The –make-mobile option ensures that all of the source code used in the translation will be packaged in the exported build session. The default mode is to only place the source code related to the issues found into the build session, which limits the ability to analyze issues completely, because only snippets are saved.

Example Translation and Export:

```
sourceanalyzer -b javaapp1   -clean
sourceanalyzer -b javaapp1  -cp  /lib/externalLib.jar:/ lib2/more.jar /src/javaapp1
sourceanalyzer -b javaapp1  -make-mobile
sourceanalyzer -b javaapp1 -export-build-session javaapp1.mbs
```

SCA Translation Log file recommendations

Always use the –debug option and the –clobber-log option to capture any issues with the translation that you may need to resolve or send to Fortify Support. You should have a process to archive log files for each scan by datetime.

NOTE: See Page 115,116 in the SCA User Guide for more details.

Example:

```
sourceanalyzer -debug -clobber-log -b javaapp1 -logfile myTrans.log < my source code directory>
```

Recommended Verify your translation.

After a translation has completed you should run the –show-files option and verify that all the files you expected to translate were translated. A file will NOT be scanned for issues, if it does NOT appear in the listing from the -show-files option.

NOTE: In the Appendix is a Shell Script example of how you could get a list of files that were not translated from the source directory. This is a demo example only.

Java Translation Examples

Command Line Translation

NOTE: For more info See pages 24-30 in the user guide for more details.

Below is a translation for an application found in the source directory /src/javaapp1 with dependency JAR files of /lib/externalLib.jar and /lib2/more.jar.

Unix/Mac OS/Linux

```
sourceanalyzer -b javaapp1   -clean
sourceanalyzer -b javaapp1  -cp  /lib/externalLib.jar:/ lib2/more.jar /src/javaapp1
sourceanalyzer -b javaapp1  -make-mobile
sourceanalyzer -b javaapp1 -export-build-session javaapp1.mbs
```

Windows

```
sourceanalyzer -b javaapp1   -clean
sourceanalyzer -b javaapp1  -cp  C:\lib\externalLib.jar;C:\ lib2\more.jar  c:\src\javaapp1
sourceanalyzer -b javaapp1  -make-mobile
sourceanalyzer -b javaapp1 -export-build-session javaapp1.mbs
```

NOTE: The java applications requires a source code list or directory and any referenced JAR files specified in the environment variable CLASSPATH or on the command line with the –cp option.

NOTE: You can use the –extdirs option to point to directories with JAR files in them.

Key Points for Java Translation

The –clean process should be done before the start of a new translation to ensure that no leftover source code that may have been removed in the development process is still in the SCA translation area.

The translation command after the clean has been performed can be called more than once if needed with the same build ID to add translation from another language type that may be part of the complete application.

Example Translation With different Language types.

```
sourceanalyzer -b javaapp1   -clean
sourceanalyzer -b javaapp1  -cp  /lib/externalLib.jar:/ lib2/more.jar /src/javaapp1
sourceanalyzer -b javaapp1  /src/htmlAndJavascriptSource
sourceanalyzer -b javaapp1  gcc Cprogram.c
sourceanalyzer -b javaapp1  -make-mobile
sourceanalyzer -b javaapp1 -export-build-session javaapp1.mbs
```

NOTE: The above commands perform translation on Java code and then appends two other language types, the translation of html/javascript code and a C program compiled with gcc.

Java Build Tool Examples

SCA can translate source code using the following build tools. There are limitations to each of the build tool support and the following should be understood.

The build tool must be able to generate a complete CLASSPATH to the required JAR files for the javac compiler locally before a SCA translation can successful.

NOTE: There are ways to use build tools in a way that works for the build process, but does not generate a CLASSPATH that the SCA translation process requires for all the source code that it is given to translate.

Maven Copy JAR files translation example

Example:

```
mvn clean
mvn dependency:copy-dependencies -DoutputDirectory=/tmpjars
sourceanalyzer -b mavenDemoWithJars -clean
sourceanalyzer -b mavenDemoWithJars -extdirs /tmpjars srcdir
```

Discussion

The solution above will use the maven copy dependencies option to download all the referenced jar file in the POM.xml to one directory and then set the JAVA CLASSPATH to that directory using the –extdirs option.

NOTE: This is a workaround for some project that may not work with the Maven Plugin. The key goal is to get all the JARs local and set the CLASSPATH for SCA translation.

This approach works when all the JARs versions referenced are at the same version in all the child pom.xml files. There are cases where some child pom.xml use an older version of the same open source library in some of the pom.xml files.

Translate Java with Maven with the Forify SCA Plugin

Solution

Install the Maven Plugin on all the build systems using Maven.

See Page 75 in the user guide for details on installing the Maven plugin.

Example:

The following commands runs the maven command to install all of the required JAR files into the Maven local .m2 repository and then translates the source code using the maven POM.xml files and the SCA Maven plugin.

```
mvn install
sourceanalyzer -b mvndemo -clean
sourceanalyzer -b mvndemo mvn clean package
```

Discussion

You must run the mvn install before using the plugin to ensure that all the JAR reference files are on the local filesystem. The Fortify SCA plugin does not support network http file paths in the CLASSPATH for translation.

The Fortify Maven Plugin using the maven copy classpath option to build the CLASSPATH for the SCA translation. Maven will fail if the dependency JAR files are not installed in the local Maven .m2 repository. Maven supports network http JAR file paths, but the Fortify SCA plugin does NOT.

Common Error Message Maven Plugin Not Installed

Example:

```
sourceanalyzer -b mvndemo mvn clean package

[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/com/fortify/sca/plugins/maven/sca-
maven-plugin/maven-metadata.xml
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.503 s
[INFO] Finished at: 2018-11-15T18:04:17-06:00
[INFO] ------------------------------------------------------------------------
[ERROR] Error resolving version for plugin 'com.fortify.sca.plugins.maven:sca-maven-plugin' from
the repositories [local (/Users/zacharylewis/.m2/repository), central
(https://repo.maven.apache.org/maven2)]: Plugin not found in any plugin repository -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following
articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/PluginVersionResolutionException
```

Common Maven Errors JARs missing from POM.xml or Maven M2 Repository

The Fortify SCA Maven plugin assumes that all reference JAR files are in the Maven local .m2 directory. All Maven referenced projects should be installed locally in the .m2 maven repository to resolve issues like the following.

Example Error Message Snippet

```
…[ERROR] Failed to execute goal on project mavendemo: Could not resolve dependencies for project com….
```

Gradle Translation

See Page 75 in the Users Guide for information on Gradle support.

NOTE: Gradle has many tasks and even custom tasks that Fortify SCA may not support.

The supported Gradle tasks for Java is the JAVA related tasks.

Known SCA Gradle Plugin JAR Reference Issue

The following error is shown with the Gradle plugin because of a bug that should have a patch soon.

```
sourceanalyzer -b g2 gradle clean build

…
TaskListener registered.
> Task :clean
[error]: Unable to resolve symbol 'Security' at
(/Users/zacharylewis/gitdirs/javatips/examples/chp3/gradle1/src/main/java/demo/program.java:10:25)
> Task :compileJava
```

Workaround for Gradle Copy Jars

A workaround is to add a tasks called ScaCopyDependencies to copy all dependency JARs to a tmp directory and then call sourceanalyzer with the –cp or –extdirs CLASSPATH setting to the tmp directory JARs.

Example Tasks.

```
task ScaCopyDependencies(type: Copy) {
  from configurations.default
  into 'scadependencies'
}
```

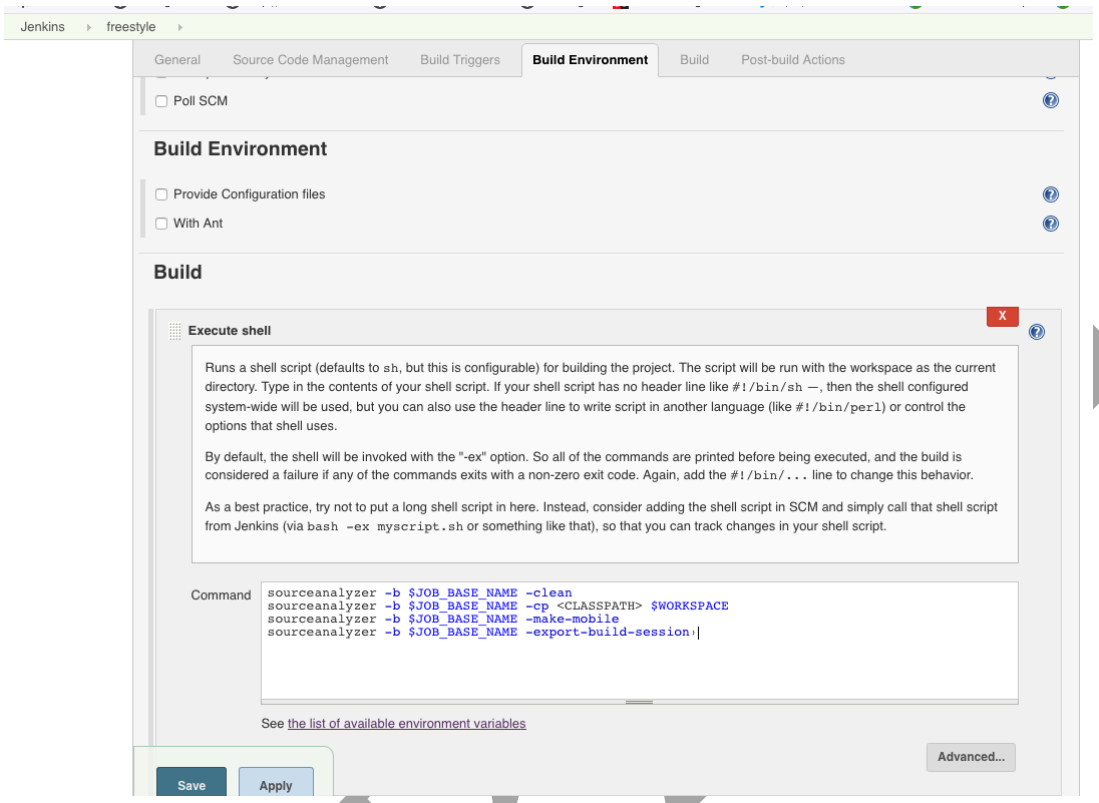
Example Workaround Commands.

The following workaround required the addeding of a tasks to copy all JAR dependency files to a directory call scadependencies and then set the classpath to point to the JARs in the scadependencies directory before running the SCA translation process.

```
gradle  ScaCopyDependencies
export CLASSPATH=scadependencies/myrefslib.jar
sourceanalyzer -b gradleworkaround  -clean
sourceanalyzer -b gradleworkaround  src
sourceanalyzer -b gradleworkaround  -make-mobile
sourceanalyzer -b gradleworkaround -export-build-session gradleworkaround.mbs
```

Java Jenkins Command Line Translation

Jenkins SCA translation can be with a command line task or using the Jenkins SCA Plugin. The recommended method is to use the command line task and to create a shell or batch script that can be reused by the applications.



Common Errors with Java Translation

The translation of JAVA source code requires that a CLASSPATH be given for all external dependency JAR files. If this is missing you will see errors similar to the following

.

Invalid CLASSPATH Error Messages:

Example: (program.java:8:31)

```
[warning]: The following references to Java functions could not be resolved. These functions may be
part of classes that could not be found, or there may be a type error at the call site of the given
function relative to the function declaration. Please ensure the Java source code can be compiled
by a Java compiler.

       getExternalFunction
```

Java Translation with Dependency JAR files.

Java source code that reference dependency JAR files must specify a CLASSPATH to all the referenced JAR files to translate completely. If the CLASSPATH is incorrect or missing then you will see error messages like the following:

Example Incorrect or missing CLASSPATH

```
Unable to locate a class for import
org.springframework.web.context.request.async.CallableProcessingInterceptorAdapter
   logger:com.fortify.frontend.translator.java.JavaResolver marker:USER thread:sourceanalyzer-13
   MDC:{class=org.springframework.samples.mvc.async.TimeoutCallableProcessingInterceptor,
frontend=JavaFrontEnd, msgId=1216, prefix=[warning]: , severity=WARNING,
sourceInfo=TimeoutCallableProcessingInterceptor.java:8:15:1, stderr=true, step=SRC_PARSE} NDC:[]
[2018-11-13 14:07:42.431 WARN  1216]
```

JSP Pages Java Related Common Errors:

JSP Pages are converted to Java code when the translation process is performed.

JSP releated Classpath Missing JAR Reference

```
[2018-11-12 09:47:56.346 WARN  12003]
Assuming Java source level to be 1.8 as it was not specified. Note that the default value may
change in future versions.
   logger:com.fortify.sca.frontend.JavaFrontEnd marker:USER thread:sourceanalyzer-13
   MDC:{frontend=JavaFrontEnd, msgId=12003, severity=WARNING, step=SRC_PARSE} NDC:[]
[2018-11-12 09:47:56.831 WARN  12022]
The class "javax.servlet.http.HttpServlet" could not be found on the classpath, but it was found in
the JAR file provided by Fortify in "C:\Program
Files\Fortify\Fortify_SCA_and_Apps_18.20\Core\default_jars\javax.servlet-api-3.0.1.jar" as a
convenience. To ensure consistent translation behavior add the JAR file that contains
"javax.servlet.http.HttpServlet" to the classpath given to the translation step. Refer to the
documentation about "default JARs" in the SCA User Guide for more information.
   logger:com.fortify.messaging.MessageManager marker:USER thread:sourceanalyzer-13
   MDC:{class=JSPPAGE._._jspXSS_jsp, frontend=JavaFrontEnd, msgId=12022, pass=Resolving,
severity=WARNING, sourceInfo=XSS.jsp:1:1, step=SRC_PARSE, webapp=C:\zacwork\nodejsdemo\demoXss\web}
NDC:[]
```

SCA Translation Environment Variables

SCA may use certain environment variables for compiled and build tools. The Build commands have certain environment variable requirements that SCA may or may not use. These will be described in the section on translation by the language type i.e "(Java, Maven, etc.)

CLASSPATH Environment Variable

SCA will use the environment variable CLASSPATH to resolve the JAR file references in JAVA and JSP translations.

Changing the default SCA Java JRE in use.

You can remove the SCA OpenJDK Java and specify a JAVA to use that is already installed. To use your installed JAVA, add the environment variable JAVA_HOME. NOTE: You must the supported Java Version Java 1.8.x.

NOTE: Fortify includes the OpenJDK 1.8 when it is installed. You can rename the SCA Installed JRE directory or remove if you want to use the local version of java that may include security updates. This SCA JRE directory is found in your installed location.

Linux/Unix/Mac OS Example

"C:\Program Files\Fortify\Fortify_SCA_and_Apps_18.20\jre\bin\java.exe" -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (Zulu 8.31.0.1-win64) (build 1.8.0_181-b02)
OpenJDK 64-Bit Server VM (Zulu 8.31.0.1-win64) (build 25.181-b02, mixed mode)

Example Windows:

set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_131

Example Unix/Linux

EXPORT JAVA_HOME=C:\Program Files\Java\jdk1.8.0_131

Appendix

Example Verify a Translation Script.

```
#
# Find all Files that could be translated by Fortify
#
# Description
#  This Script will read the fortify sca properties file and find all the supported types and
#  Search the directory for all files that match supported file types by extension and
#  compare against the Build ID List of file translated
#
function checktranslation()
{
  #echo "DEBUG: checking: $fileext"
  for filesrc in `find .  -name "*${fileext}"`
  do
   basepath=`echo $filesrc | cut -c3-`
   grep $basepath  ${TMPFILE} > foo
   if [ $? = 0 ];
   then
    foo=foo
   else
    echo "SCA Source File Not Translates: ($filesrc)"
   fi
  done
}
TMPFILE="`pwd`/scaTranslated.tmp"
SCACMD=`which sourceanalyzer `
SCAConfigDir=`echo ${SCACMD}  | sed 's;bin/sourceanalyzer;Core/config;'`
FortifyConfigFile="${SCAConfigDir}/fortify-sca.properties"
BUILDID=$1
SOURCEDIR=$2
if [ $# -gt 1 ];
then
 if [ ! -d ${SOURCEDIR} ];
 then
  echo "ERROR: Source not found ! Verify it exists and is readable"
  exit 1
 fi
 if [ -r "$FortifyConfigFile" ];
 then
  $SCACMD -b ${BUILDID} -show-files > ${TMPFILE}
  cd ${SOURCEDIR}
  # Set File Extension Type for supported Translation languages.
  scafileext=`grep com.fortify.sca.DefaultFileTypes $FortifyConfigFile`
  cnt=1
  fileext=`echo  ${scafileext} | cut -d= -f2 |  cut -d, -f1 `
  while true
  do
   cnt=`expr $cnt + 1`
   if [ "$fileext" != "" ];
   then
      # Find All potential files that can be translated
    checktranslation
    fileext=`echo  ${scafileext} | cut -d\, -f${cnt}`
   else
```

```
    exit 0
   fi
  done
 else
  echo "ERROR:  Fortify Config File not Found at: ($FortifyConfigFile)"
  echo "Verify that sourceanalyzer is install and in your PATH . "
  exit 2
 fi
else
   echo "Usage:  verifyTranalation.sh <SCA BUILD ID> <Source Code Dir>"
   exit 99
fi
```

Example Simple Java Translation Script

```
#
# This Simple Script to translate java source code by creating the CLASSPATH from all
# JARs files found in the source code directory tree.
#
#

SCAVALID=`which sourceanalyzer`
if [ $? -ne 0 ];
then
  echo "ERROR: sourceanalyzer is missing from your bin PATH directories."
  echo "FIX: set the PATH to the location of the installed bin directory for sourceanalyzer"
  exit 99
fi

if [ $# -gt 1 ];
then

  BUILD_ID=$1
  SOURCE_DIR=$2

  CLASSPATH=${SOURCE_DIR}:
  for jarfile in `find ${SOURCE_DIR} -name "*.jar" `
  do
    CLASSPATH="${CLASSPATH}:${jarfile}:"
  done

  sourceanalyzer -b ${BUILD_ID} -clean
  sourceanalyzer -b ${BUILD_ID} -verbose -logfile ${BUILD_ID}.log -debug -cp ${CLASSPATH}
${SOURCE_DIR}
  sourceanalyzer -b ${BUILD_ID} -make-mobile
  sourceanalyzer -b ${BUILD_ID} -export-build-session ${BUILD_ID}.mbs

else

  echo "usage translateJava.sh <BUILD ID> <Source Directory>"
  exit 1
fi
```