Date: 30 Jul. 14

Author: Jasper de Keijzer

About this doc 1

Security Warnings the old way 1

Other options 2

Rule set and no warnings at all 3

How to stop the security warnings for a known applet 3

Step 1: Create a key in your keystore 3

Step 2 : Create a ruleset.XML 4

Step 3: Create JAR’s 4

Step 4: Sign the JARS with the keystore 5

Step 5: Copy the DeploymentRuleSet jar to the java deployment 5

Step 6: Generate a certificate 5

Step 7: Import the certificate in the Java Console 5

Step 8: Creating the HTML file with the applet 6

Step 9 : Running the applet in UNIFACE 6

Resume 7

## About this doc

Security warnings are hindering the end user when starting a Java applet in the UNIFACE html widget. This doc describes how to stop the security warning and even block them with a so called “rule set”.

## Security Warnings the old way

According to the documentation of Oracle, the end user will in almost all cases be presented with a warning when starting a Java applet in the browser for the first time. Even the lowest possible security setting in the Java console explains:

Medium - All applications are allowed to run with security **prompts**.

See <http://docs.oracle.com/javase/7/docs/technotes/guides/jweb/jcp/jcp.html>

Also the list of exceptions shows in the java console that you can be prompted by a security warning:



By switching the cache ON in the Java console the warning is only displayed once. After this the application runs without warnings and can even be re-started.



## Other options

Keep the security settings in the Java console on High which by default blocks the applet completely.



Add the URL to the list of exceptions:



In my case this was:

[file:///D:\usys91\HTML\_JavaApplet\mx04\dynamic\applet\_ComponentArch\_DynamicTreeDemo\build\classes\AppletPage\_WithAppletTagUsingJNLP.html](file:///D%3A%5Cusys91%5CHTML_JavaApplet%5Cmx04%5Cdynamic%5Capplet_ComponentArch_DynamicTreeDemo%5Cbuild%5Cclasses%5CAppletPage_WithAppletTagUsingJNLP.html)

Including the page name!

This means that the security is not compromised and the warning is only shown once when the cache is on.

## Rule set and no warnings at all

As explained earlier in this document, you can run a java applet without security warnings by using a rule set however the applet must be signed for this and a so called deployment rule set jar file must be added. In the following places you can find some documentation. In the next chapters I describe a step by step process to get the Java applet running in a UNIFACE html widget without warnings.

<https://blogs.oracle.com/java-platform-group/entry/introducing_deployment_rule_sets>

<http://wraithhacker.com/java-deployment-rule-set/>

# How to stop the security warnings for a known applet

In the following chapters I will take a step by step approach to make it possible to run a known applet in the UNIFACE html widget without bothering the end user with security warnings. The applet JAR file, in this sample, is on the end user computer as well as the html file referring to this applet. Of course the file:/// can be replaced by a server site location like http://

This small manual on how to get the “rule set” working is based on the Dynamictree sample of Oracle. You can find this sample on the following address:

<http://docs.oracle.com/javase/tutorial/deployment/applet/examplesIndex.html>

Download the zip file with all the bits and pieces you need:

<http://docs.oracle.com/javase/tutorial/deployment/applet/examples/zipfiles/applet_ComponentArch_DynamicTreeDemo.zip>

Before you start, be sure that you have your path variable set to the java bin folder otherwise the command lines shown in the steps won’t work.

Used command line tools

|  |  |
| --- | --- |
| Command | Description |
| Jar | Creates a Jar archive |
| Keytool | To create a keystore, certificate  |
| Jarsigner | To sign a jar archive with a keystore file. |

## Step 1: Create a key in your keystore

The first step is to create a keystore file named Mykeystore in the current working folder. I use a password “java\_woody” but this can be anything you want as long it is more than 5 characters long. I use the key algorithm EC since this can be loaded into the Java console. RSA does not work correct. The sizes of the different key algorithms can be found on the Oracle Java sites.

keytool -genkey -keystore Mykeystore -keypass java\_woody -alias selfsigned -keyalg EC -keysize 571 -validity 10000

## Step 2 : Create a ruleset.XML

In this step we create a rule set xml file describing the rules for our applet. In this file “ruleset.xml” you specify where the html file is located which contains the applet and what to do with the applet. I also specify the java version to use. You can leave this out. The second rule takes care that there can be no other applet started and will show the message “Blocked by corporate Jasper”.

<ruleset version="1.0+">

<rule>
<id location="file:///D:/classes/AppletPage\_WithAppletTagUsingJNLP.html" /> <action permission="run" version="1.7.0\_65" />
</rule>

<rule>
<id />
<action permission="block">
<message>Blocked by corporate Jasper</message>
</action>
</rule>

</ruleset>

The last rule causes the appearance of the following message when starting an applet which is not in the rule set.



Remove the last rule if you want to run all other applets under the default security settings.

## Step 3: Create JAR’s

In this step we will create two jar files. One containing the Java classes for the applet and its manifest file and one file containing a the ruleset.xml file. The last one must be named: “DeploymentRuleSet.jar”.

Create the jar from the java class files placed in the sub folder named “appletComponentArch”.

Jar cvfm DynamicTree.jar manifest.mf -C .\appletComponentArch\\*

Create a jar file from the ruleset.xml and name it “DeplymentRuleSet.jar”.

jar cvf DeploymentRuleSet.jar ruleset.xml

## Step 4: Sign the JARS with the keystore

Both jar files should now be signed with the keystore Mykeystore using the alias selfsigned. This name “selfsigned” could be any name, I choose this to show that it is a selfsigned jar. The store pass is the password on the keystore something you can change.

jarsigner -keystore Mykeystore -storepass java\_woody -keypass java\_woody DynamicTree.jar selfsigned

jarsigner -keystore Mykeystore -storepass java\_woody -keypass java\_woody DeploymentRuleSet.jar selfsigned

Both jar files are now signed with the same key stored in the keystore under the alias “selfsigned”.

## Step 5: Copy the DeploymentRuleSet jar to the java deployment

Copy the DeploymentRuleSet.jar to C:\Windows\Sun\Java\Deployment

Java runtime looks into this folder for the DeploymentRuleSet.jar.

## Step 6: Generate a certificate

In this step you generate a certificate from the keystore named Mykeystore and redirect the output to the file named DynamicTree.csr. Csr is the default extension used by the Java console.

keytool -certreq -keystore Mykeystore -storepass java\_woody -alias selfsigned >DynamicTree.csr

This file needs to be imported in the Java console.

## Step 7: Import the certificate in the Java Console

The certificate generated in step 6 can now be loaded in the Java Console. See in you start menu on Windows



Or open the java console from the java bin folder: javacpl.exe

On the security tab press:



Select “Signer CA” from the drop down list:



Import the DynamicTree.csr you created in step 6.

Close the console and just to be sure you load the correct applet, switch the cache off for the moment. Later you can switch it on again for performance reasons.

## Step 8: Creating the HTML file with the applet

This file comes with the Oracle sample. Also the referenced 'dynamictree\_applet.jnlp'file. Be sure it is in the right spot. I changed some naming and layout to fit it in a screen shot used later in this doc.

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><html lang="en-US"> <head> <title>UNIFACE Applet Page</title> <meta http-equiv="Content-Type" content="text/html; charset=windows-1252"> </head> <body> <noscript>JavaScript enabled is required for this page to operate properly.</noscript> <h2>UNIFACE Applet demo</h2> <applet alt = "Dynamic Tree Applet Demo" code = 'appletComponentArch.DynamicTreeApplet' jnlp\_href = 'dynamictree\_applet.jnlp', width = 300, height = 300 /> </body></html> |

Save this file in the location which was defined in the ruleset.xml

Name: AppletPage\_WithAppletTagUsingJNLP.html

Location: D:/classes/

## Step 9 : Running the applet in UNIFACE

Draw the html widget on a form and call the widget operation

$fieldhandle(html.html)->$widgetoperation("loadURL","file:/// D:/classes/AppletPage\_WithAppletTagUsingJNLP.html")

Be aware that the location **MUST** be the same as defined in the ruleset.xml.



To see whether I ran the correct applet I changed the Root Node into Root UNode.

## Resume

1. Get the zip file containing the tree sample from the Oracle site.
2. Keystore created with the correct key algorithm.
3. Create the ruleset.xml
4. Make the JAR files with the JAR tool for both the applet and the ruleset.xml
5. Signe the jar files using our key store.
6. Copy the DeploymentRuleSet.jar to sun/java/deployment folder
7. Generate a certificate from the keystore with the keytool
8. Imported the generated certificate into the Java consol under “Signer CA”
9. Created an HTML document in the right spot and the right name referring to our applet.
10. In the HTML widget we loaded the HTML document with “loadurl” widget operation.

Success!