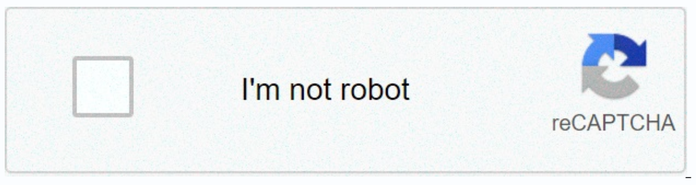

Continue

Java security site exception list

I need to use an old Java applet for a particular website, but newer Java versions can't run it, because it has a self-signed certificate. Reading on Oracle's Distribution Guide, I need to do my own distribution. JAR, with a proper certificate signing (not self signed), just to create the exception I need to run applets from a single domain. I have found that there is a %userprofile%\appdata\LocalLow\Sun\Java\Deployment\security\exception.sites file, apparently one for each user. In there, one per line, is a list of domains I can set up to exclude from NOT being performed. When I add the domain name, the user receives a single prompt, then

it only allows the domain for that user. Large. I know I could add this file to the Default User Profile, for all new users that are created. Unfortunately, I don't think removing each user's profile from each of the systems is the way to go. I can push commands to computers, but the commands run as my user - I can't use %USERPROFILE% for this. I don't want to put it in my login script, as I don't want the file to be large for users who are constantly logging in and off of systems. I also want only the only domain added to what is available, without obliterating the user's preferences that may already exist. Since I don't want to obliterate their preferences, I thought I'd do something like echo >> %userprofile%\appdata\LocalLow\Sun\Java\Deployment\security\exception.sites, but that won't work, because it will continue to add to the file. What can I do to add the list to the Exceptions rule, but only if the rule doesn't already exist? 18 Sep 2018 Since java Runtime 7 Update 51 release applications must now meet the requirements for security settings High or Very High, or be part of the exception site list before they can run. Some some government website here in Australia still requires Java Runtime for login, such as tax agent portal and AUSKey Authentication Services – and it seems that they haven't really gotten on top of keeping their certificates up to date. While tt would be good to debate about the importance of certificates, when you have a company with 30 accountants screaming that they can't log in then unfortunately exceptions (pun intended) have to be made. In fact, it is the recommended fix of the self-proclaimed Technical Help Desk for Australia. Java Runtime stores it's exception list in a common text file,exception.sites, which is located in %userprofile%\AppData\LocalLow\Sun\Java\Deployment\security - the file format is one location per line. Copy / create the exception.sites file to a network location. Create or edit a GPO in the most appropriate organizational unit Navigate through GPO to Computer Configuration > Preference > Windows Settings > Files > ... Configure the following options: Action: Create source file: Destination file: %userprofile%\AppData\LocalLow\Sun\Java\Deployment\security\exception.sites Revel in the sadness that you are is encourage a governmental organisation not to keep track of the expiry of its certificates. The location of the exception site list is specified in the deployment.user.security.exception.sites property. The default location is <deployment.user.home>/security/exception.sites. See Chapter 21, Configuration File, and Deployment Properties for information about properties and property files. Users can manage a list of their systems, or use a list managed by a system administrator in a central location. If a system administrator does not want users to edit the exception site list, the deployment.user.security.exception.sites can be set to a file for which users do not have write permission. If a user cannot write to the exception location list, the list appears in Control Panel for Java, but the edit controls are not available in the Exception Site List window. To prevent users from using an exception site list other than the list set up by a system administrator, the deployment.user.security.exception.sites property can be locked. See Section 21.2, Deployment Configuration Properties for system property locking information. Page 2Topics in this section provide additional information for deploying Java and JavaFX applications. Page 3 Hypertext Transfer Protocol -- HTTP 1.1 (RFC 2616) discusses HTTP compression. HTTP compression allows application JAR files to be deployed as compressed JAR files. The supported compression techniques are gzip, compress and deflate. Starting with JDK version 5.0, HTTP compression is implemented in Java Web Start and Java Plug-in in accordance with RFC 2616. The supported technologies are gzip and pack200-gzip. The requesting application can send an HTTP request to the server that indicates its ability to handle compressed versions of the file. The following example shows the HTTP request that is created when the Dynamic Tree Demonstration applet, whose JAR file has been compressed with Pack200, loads: The following example shows the HTTP response from the server: For more information about the dynamic tree demopp, see Deploy an Applet in java tutorial. The Accept Encoding field specifies what the client can accept, as specified by the client. The Content-Encoding field specifies what is sent, as specified by the server. The Content Type field specifies what the client should expect when the transformation or decoding takes place. In this example, the Accept-Coding field is set to pack200-gzip and gzip, indicating to the server that the application (in this case Mozilla Firefox running in Windows 7 with the Java Plug-in supplied with JRE 7) can handle the pack200-gzip and gzip formats. The server searches for the requested JAR file with the .pack.gz or .gz extension and responds with the specified file. The server sets the Content Encoding answer header to pack200-gzip, gzip, or NULL depending on the type of file being sent, and may set the Content-Type to application/x-java-archive. By inspecting the Content-Encoding field, the requesting application can apply the corresponding transformation to restore the original JAR file. In Figure 30-1, the client requests the foo.jar file with the Accept-Encoding200-gzip,gzip field package. The server searches for the foo.jar.pack.gz file. If the server finds the file, it sends the file to the client and sets the Content Encoding field to pack200-gzip. In Figure 30-2, if the file foo.jar.pack.gz is not found, the server responds with the file foo.jar.gz, if it is found, and sets the Content-Encoding field to gzip. In Figure 30-3, if the files foo.jar.pack.gz and foo.jar.gz are not found, then the server responds with the file foo.jar and either does not set the Content-Encoding field or sets it to NULL. In Figure 30-4, an older program (a program without HTTP or Pack200 compressions) requests the foo.jar file; consequently this application will continue to work seamlessly. Therefore, it is recommended that you host all three files foo.jar, foo.jar.gz, and foo.jar.jar.gz. Page 4 In a JNLP file, the codebase is an optional parameter to the <jnlp>tag. It is used both to locate the resources described in the JNLP file, as well as to find the JNLP file itself. For technical reasons, Java Web Start cannot update the contents of the JNLP file from the server unless an absolute code base is specified. A code base is always provided by the browser, either because it was explicitly specified, or implicitly obtained from the location of the HTML document. This allows relative URLs to be used in JNLP files, which is very useful for moving an entire content tree from one server to another. JNLP files refer to other JNLP files in a tree structure. The root JNLP file for a JNLP applet is referenced by a <applet>tag. Applet tag's codebase helps define the location of the root JNLP file. The code base calculation rules are as follows: If an absolute code base is specified in the JNLP file, it is used. This is required for backward compatibility reasons. If the JNLP codebase is not specified, the directory that contains the JNLP file is used. Otherwise, merge jnlp's codebase into the directory that contains the JNLP file. In simple Java terms, this can be expressed as shown in the following example: URL new_codebase = new url(current_jnlp_dir, current_jnlp_codebase); This codebase calculation is not an extension of the JSR-56. The JSR-56 does not limit the codebase to absolute, and therefore it can be relative. Example 1: this JNLP's location: this JNLP's codebase: resulting codebase for parsing this JNLP: Example 2: this JNLP's location: resulting codebase for parsing this JNLP: Example 2: this JNLP's location: <5> this JNLP's codebase: resulting codebase for <none>parsing this JNLP: Example 3: this JNLP's location: denna JNLP: s kodbas: codebasedir resulterande</none> </applet> </jnlp> </jnlp> for interpreting this JNLP: example 4: Relative paths are used to refer to each nested JNLP, just like in a tree with HTML files. www.example.com/html/my_applet.html refers to: my_applet.jnlp codebase: www.example.com/html my_applet.jnlp: codebase not specified inherits www.example.com/html references JNLP extension jogl/jogl.jnlp jogl/ jogl.jnlp codebase not specified inherits www.example.com/html/jogl (the directory containing jogl.j nlp) references gluegen-rt/gluegen-rt.jnlp gluegen-rt/ gluegen-rt.jnlp codebase not specified inherits www.example.com/html/jogl/gluegen-rt (the directory containing gluegen-rt.jnlp) Page 5 Earlier, when a new JRE was installed, JAR and native libraries for the old edition were not visible to the new one. Thus, applications and applets that relied on these libraries would not work, and developers had to relocate their libraries to the new JRE. The System-Wide Repository is the solution to that problem. It provides a global or system-wide location where the VM can search for libraries, JAR, or native speakers, regardless of the JRE site. It provides features similar to the Microsoft VM repository. Note: The system-wide repository is supported by the Java Plug-in only on Windows. The table below shows the location of the new System-Wide Repository: System-Wide Site DLL Type in the System Path environment variable. Untrusted JAR <Windows directory=>\Sun\Java\Deployment\Lib\Untrusted JAR <Windows directory=>\Sun\Java\Deployment\Lib\Trusted <Windows directory=>is the Windows OS directory on the drive where Windows was installed (also known as %SystemRoot%). For example, on Windows 7, where Windows is installed on the C drive (typical), the locations of these libraries would be as follows: Type of library System-Wide Location DLL Any directory in the System Path variable; for example, C:\WINDOWS\repository if C:\WINDOWS\repository is set in the System Path variable. Untrusted JAR C:\WINDOWS\Sun\Deployment\Deployment\Lib\Untrusted JAR C:\WINDOWS\Sun\Deployment\Lib\Trusted The classes in the JAR files from the system-wide trusted repository are loaded by the add-on class loader, while the classes in the JAR files from the system-wide untrusted repository are loaded by the applet class class. Thus, the previous classes are given the AllPermission permissions, while the latter are given only standard applet permissions. The system-wide trusted repository implementation is based on the java.ext.dirs system property. If users select their own java.ext.dirs system property through the Java control panel, the JAR files in the system-wide trusted repository will not be loaded by java plug-in. The database does not provide version or namespace control. It is up to the deployer to avoid version and name spaces conflicts at deployment time. Page 6 By default, the JavaFX application proxy settings are taken from aktuella webbläsaren om</Windows> </Windows> </Windows> </Windows> is embedded in a web page or system proxy settings are used. Sometimes SOCKS proxy settings are ignored, as specified in System.setProxy. If you need to disable the automatic proxy configuration in the application, enter a JavaFX-Feature-Proxy manifest entry in fx:jar with a value of None as in the following example. <manifest> <attribute name=JavaFX-Feature-Proxy value=None></attribute> </manifest> After you enter the JavaFX-Feature-Proxy manifest, the network stack will not be initialized before the application code gets executed and you can set socks properties in the code. Page 7 The following improvements have been made to the JNLP file syntax; see JNLP File Syntax: The OS attribute in the information and resources elements can now contain specific versions of Windows, such as Windows Vista or Windows 7. Applications can use the install attribute in the shortcut element to indicate their desire to be installed. Installed applications are not removed when clearing the Java Web Start cache, but can be explicitly deleted using the Java control panel. Java Web Start applications can be deployed without specifying the codebase attribute; see Deploy without Codebase A JNLP file can be embedded into an HTML page; see Embedding JNLP File in Applet Tag. You can check the applet's status variable while it is loading to determine if the applet is ready to handle requests from JavaScript code; see Site management status with event handlers. You now have control over the window decoration style and title of an applet launched from a shortcut or pulled out of the browser; see Request and customize Applet decoration in Develop draggable Applets. To make it easier to redeploy an application that is deployed using a signed jar file, a JNLP template can be used. The JNLP template allows the developer to specify which parts of a JNLP file can be modified without the main jar departing. For more information, see Section 17.2, Sign a JAR file with a JNLP template. Additionally, the following improvements are introduced: Area: DeploymentStandard/Platform: JDK 7Synopsis: Previously, the pack200 tool segmented its output by default. Starting with this release, the pack200 tool will create a large segment per jar file. Therefore, if deployers have jar files larger than the virtual memory that is on the end user's system, it is recommended that either the input jar jar file be split, or appropriately segmented by using the --segment-limit-nnnnnn command-line flag or the corresponding property SEGMENT_LIMIT. RFE: 6575357 Area: DeploymentStandard/Platform: JDK 7Synopsis: On Windows XP machines, the default cache directory is now under $USER\Local Settings\Application Data\Sun\Java\Deployment\cache. If necessary, you can customize it to point to a network resource folder for a single application cache of domain machines. RFE: 7012538 Area: DeploymentStandard/Platform: JDK 7Synopsis: Optional applet parameter, jnlp_embedded, provides the option to cache JNLP content on the HTML page and shorten the applet launch time by skipping network access. The parameter jnlp_embedded as its base64-encoded content in the applet JNLP file. For example: <applet width=710 height=540> <param name=jnlp_href value=launch.jnlp> <param name=jnlp_embedded value=PD94bWwgdmVyc2lvbj0iMS4wIiB . . dC1kZXNjPg0KPC9qbmxwPg0K> <param name=draggable value=true> </applet> When present, jnlp_embedded parameter value replaces the contents of the JNLP value pointed to by the jn.p_href parameter. The value of jnlp_ref is then optional and is used only as a backup when the contents of the jnlp_embedded is invalid. There are some limitations to the embedded JNLP content: The href attribute from the jnlp element should be relative. The codebase attribute from the jnlp element should be empty (meaning that the code base value will be derived from the document base URL). RFE: 6990877 Area: DeploymentStandard/Platform: JDK 7Synopsis: Starting with JDK 7, clearing the cache works as follows: Only uninstalled resources are removed from the cache when javaws -XClearCache is invoked. Both installed and uninstalled resources are removed by invoking javaws -uninstall. The Delete Temporary Files dialog box in JCP previously had two check boxes, both checked by default: Applications and Applets, and Trace and Log files. Starting with JDK 7, there will be three checkboxes, with the first two checked by default: Trace and Log Files, Cached Applications and Applets and Installed Applications and Applets. Only installed applications will have an entry in the add-on/remove panel. RFE: 6873615 Area: JNLP FilesStandard/Platform: JDK 7Synopsis: To support troubleshooting Java Web Start applications, the -XX:HeapDumpOnOutOfMemoryError flag is now supported in JNLP files for trusted applications. RFE: 6664424 Area: JNLP FilesStandard/Platform: JDK 7Synopsis: Previous versions of Java Web Start did not properly implement section 6.0.10 of the JNLP Specification. With this fix, fine-grained values for the os attribute, such as os=Windows\ XP, os=Windows\ Vista, and os=Windows\ 7, will work as expected. Values such as os=Win, and os=Windows will continue to match all Windows platforms. As of this release, os =Windows\ Vista Windows\ 7 will only match Vista or Windows 7 and not Windows XP. RFE: 7014170 Area: PluginStandard/Platform: JDK 7Synopsis: A message that the first generation of Java plugin is overdue is now printed to the log file and to the Java console when plugin 1 is used. RFE: 7027792 Area: PluginStandard/Platform: JDK 7Synopsis: On Windows, the tile icon is now disabled by default. To activate the tray icon, use the Windows Start menu. RFE: 6694710 Area: PluginStandard/Platform: JDK 7Synopsis: The 64-bit toolkit is now supported on 64-bit Windows platforms. RFE: 6492139 Area: JDK 7Synopsis: Previously, the persistence API only gave temporary persistence for When the VM was terminated, the data was lost. Data may persist through any of the usual Java mechanisms, through the use of LiveConnect to store data in the DOM page, or through one of the JNLP services. RFE: 6992419 Area: PluginStandard/Platform: JDK 7Synopsis: Google Chrome is now supported by DT Plugin.RFE: 6907245 Page 8 The Java Runtime Environment Settings have two tabs, Users and System. Both tabs display a table that contains information on the JRes installed on your system. The Users tab shows all registered JVs and JVS that the user has added. The System tab shows the jre that was used to start Control Panel for Java. Each row in the table represents a Java Runtime Environment (JRE) installed on the computer. The following information is provided for each JRE: Platform: The version of the JRE product: The full version number of jre, including the update number Location: the URL that the Java Update Scheduler uses to launch automatic updates Path: The full path to the JRE Runtime parameters: Optional custom options used to override the Java Plug-in default boot parameters Enabled: Flag indicating which of the JRE versions is considered when running an application with Java Plug-in or Java Web Start. Settings in Java Control Panel do not apply to standalone or self-contained applications. If the check box for a JRE is not selected, then Java Plug-in and Java Web Start will not use JRE to launch Java applications. However, the current JRE may be used even if it is not marked as enabled. Note: If Java content in the browser is disabled in the Security tab of the Java Control Panel, it has no effect to enable JRE in the Run Time Java Environment Settings dialog box. See section 20.4, Security for information about the Enable Java content option in your browser. To change the information for a JRE in the Users tab, click the cell in the table and edit the value. The information in the System tab cannot be edited. The Users tab provides the following features: Click Search to start JRE Finder. Use this tool to search for JRE installed on your computer and add them to the table. Click Add to manually add a JRE to the table. A new row is added to the table. Fill in the values for Platform, Product, Path, Runtime Parameters, and Enabled. Click Delete to remove the selected JRE from the table. The table always has at least one record, which is the last jre installed. This is jre associated with the Java control panel. Microsoft Windows displays all JRE's installed on a computer. The Java control panel finds the JRE's by looking in the registry. On Solaris, Linux and OS X, JRE that Java Web Start or Java Plug-in uses to deploy applications is the JRE that is considered to be registered. Therefore, use the Search, Add, and Remove buttons to which JREs are included in the table. On OS X, only the currently installed JRE is displayed, JDKs are not For Solaris, Linux, or OS X, only version 5.0 or higher should be added. For Microsoft Windows, where all JRE is in the registry, version 1.3.1 or higher is displayed. The 9A deployment.properties user-level file is always available. Its location, which is not configurable, is described in user level. There may also be an optional system-level deployment.properties file. If it exists, its location is determined by a System Administrator through the deployment.config file, as described in system level. The following table lists the location of the user-level distribution.properties file. Table 21-1 User-level deployment configuration File Operating System Location Windows <User application= data= folder=>\LocalLow\Sun\Java\Deployment\deployment.properties Solaris, Linux ${user.home}/java/deployment/deployment.properties OS X ~/Library/Application Support/Oracle/Java/Deployment/deployment.properties On Windows, <User application= data= folder=>normal Is C:\User\User\User On Solaris and Linux, ${user.home} is typically /home/username. On OS X, the tildet (~) represents the home directory, which is typically /Users/user name. The following sites contain examples for each operating system: For user jed running on Windows 7, the deployment.properties file would be in the following directory: C:\Users\jsmith\AppData\LocalLow\Sun\Java\Deployment\deployment.properties For user bjones running on Solaris or Linux, deployment.properties file would be in the following directory: /home/bjones/.java/deployment/deployment.properties For users jdoe running on OS X, the deployment.properties file would be in the following directory: /Users/jdoe/Library/Application Support/Oracle/Java/Deployment/deployment.deployment.the deployment.config file is used to specify system-level deployment.properties in the infrastructure. By default, there is no deployment.config file, so there is no system-wide deployment.properties file. If the deployment.config file exists, it is in one of the directories listed in the following table. Table 21-2 Site of system-level deployment configuration File operating system location Windows <Windows directory=>\Sun\Java\Deployment\deployment.config ${deployment.java.home}\lib\deployment.config Solaris, Linux /etc/.java/deployment/deployment.config ${deployment.java.home}/lib/deployment.config OS X /Library/Application Support/Oracle/Java/Deployment/deployment.config ${deployment.java.home}/lib/deploy/deployment.config ${deployment.java.home} is the site of the JRE from which the distribution products are run. Distribution products include Java Web Start, Java Plug-in, Java Control Panel, and others. The deployment.config file contains two properties: deployment.system.config and deployment.system.config.required. The deployment.system.config property is the URL of the system (enterprise-wide) deployment.properties file. This property can be used by system administrators to centrally</Windows> </User> or user-specific lock-down configuration settings. For local files, use the file protocol in the URL, such as file:///C:/Windows/Sun/Java/Deployment/deployment.properties. Note: If the format of the file protocol shown in the example does not work for you, try one of the following alternate formats: file\\\\C\\deployment.system.properties file:\\C:\\deployment.system.properties file:/\deployment.properties. If it is set to true, the deployment.properties file pointed to by the deployment.system.config property must be found and successfully loaded, otherwise nothing is allowed to run. If the property is set to false, the deployment is attempted to locate and load. properties file pointed to by the deployment.system.config property. If successful, the file is used, otherwise the file will be ignored. The default for the deployment.system.config.mandatory property is false. Page 10 Trace for Java Plug-in and Java Web Start can be turned on by setting the deployment.trace property to true. This property turns on all tracking devices inside the Java Plug-in and Java Web Start. To enable more fine-grained tracking, the deployment.trace.level property can be used. Deployment.trace.level property can have one of the following values: basic cache-net security ext liveconnect drs To use all trace levels, set deployment.trace.level to everyone. To enable tracking while running, you can set options at the tracking level (0-5) in the Java console, with the following meanings: 0: off 1: basic 2: network, cache and basic 3: security, network and basic 4: extension, security, network and basic 5: LiveConnect, extension, security, network, temp, basic and distribution rule set See Section 22.1, Debugging Options in the Java Console for information. Another way to set fine-grained tracking is through the Java Control Panel. For example, to enable tracking for everything (option 5 above), enter the following options in the Java Run Time Parameters text field: -Ddeployment.trace=true -Ddeployment.trace.level=all See section 20.3.2, Java Runtime Parameters for more information. Tracking set through Control Panel takes effect when Java Plug-in or Java Web Start is started, but changes made through Control Panel while Java Plug-in or Java Web Start are running have no effect until restarted. Page 11Topics in the section provide information about the security features available to your Java and JavaFX applications. Page 12Beginning with the 7u21 release, users are notified when a RIA is launched with a security query similar to the following screenshot. Depending on RIA, the security information displays the following information: NAME of RIA, or message that the application is unsigned. The name shown is the value of the Application-Name attribute in About About attribute is not present, the value of the Main-Class attribute is used. If none of the attributes are in the manifest, the security prompts do not display a title. Titles are not displayed for unsigned RIA. See section 26.3, Application-Name Attribute for details. Warning when an outdated JRE is used. For companies that manage the update process for the user's system, deployment property deployment.expiration.check.enabled may be set to suppress warnings for in-of-date JREs. See Chapter 21, Configuration file, and deployment properties for information. Information about the publisher. If the application is self-signed or signed by an unknown authority, the publisher appears as UNKNOWN. Certificate warnings. If the certificate has expired, revoked, or the server that tracks which certificates were revoked cannot be accessed, the prompt displays a warning. A warning is also displayed if the certificate is not valid until a future date. Location from which the application is accessed. This value is either a URL for applications accessed from a Web site, or a directory for applications accessed from a local drive. Level of access required by the application. Restricted access restricts the application to the security sandbox, unrestricted access gives the application access to resources on the user's system. Missing JAR file manifest attributes warning. JAR file manifest attributes are available to provide additional protection for an application. The warning indicates that the manifest is missing a recommended attribute. See Chapter 26, JAR File Manifest Attributes for security for information. For unsigned or self-signed applications, a check box that the user must select before the Run button is activated. Ability not to display the prompt again. For signed RIA, future prompts for this RIA and RIA from the same location that are signed with the same certificate may be turned off. If RIA is a sandbox application, then the prompt is turned off only for other sandbox applications that are signed with the same certificate. When you appear, click Show Options to access the option to turn off the prompt. Prompts that were previously hidden can be recovered through a button on the Security tab of the Java Control Panel. See section 20.4.4, Reset security information for information. Buttons to run RIA, Cancel RIA, and when a JRE is entering today, to Update to the latest JRE. For a description and example of security corruption, see What should I do when I see a java recovery screen? on java.com.Page 13 The last step in the deployment process is to find out if the RIA files are running, and answers the question in the last step of the deployment process, What way is needed? If RIA is running, this step also determines which version of JRE is used when an older version is requested. The Normal Processing and Exception List box in Figure 24-1 shows the process of determining RIA is running and which security prompt is displayed. These boxes are connected to the node for this step and appear to the right of the node. The process of determining which JRE is used when requesting an older version is displayed is shown in the Select JRE box in Figure 24-1, which is connected to the boxes to determine whether RIA is running and which security prompt is displayed. At this point in the process, the decision to run or block RIA, and the choice of security query to view, is based on the following criteria: The setting of the security level in Java Control Panel, default is High. See section 20.4, Chapter 20 Security, Java Control Panel for information about the security level. The presence of a RIA Certificate Signing Certificate that was used to sign RIA The introduction of the RIA site in the exception location list Note: If the user previously selected the option not to view a query again, then the prompt is suppressed. The response the user gave when the option to suppress was selected is used instead of displaying the prompt again. Page 14 Use jarsigner to sign the JAR file, using the credentials in your keystore that were generated in the previous steps. Make sure that the same alias name is specified, for example: C:\Program Files\Java\jdk1.8.0\bin\jarsigner C:\TestApplet.jar MyCert Enter Passphrase for keystore: ******** Use jarsigner -verify -verbose -certs to verify jar files. C:\Program Files\Java\jdk1.8.0\bin\jarsigner -verify -verbose -certs d:\TestApplet.jar 245 Wed Mar 10 11:48:52 PST 2000 META-INF/manifest.mf 187 Wed Mar 10 11:48:52 PST 2000 META-INF/MYCERT. SF 968 Bag Mar 10 11:48:52 PST 2000 META-INF/MYCERT. RSA smk 943 Wed Mar 10 11:48:52 PST 2000 TestApplet.class smk 163 Wed Mar 10 11:48:52 PST 2000 TestHelper.class X.509, CN=XXXXXXX YYY, OU=Example Software, O=New Technology Company, L=Cupertino, ST=CA, C=US (mycert) X.509, CN=New Technology Company, OU=Java Plug-in QA, O=New Technology Company, L=Cupertino, ST=CA, C=US X.509, EmailAddress=server-certs@thawte.com, CN=Thawte Server CA, OU=Certification Services Division, O=Thawte Consulting cc, L=Cape, ST=Western Cape, C=ZA s = signature checked m = entry listed in manifest k = at least one certificate found in keystore i = at least one certificate found in identity jar scope verified. Your RIA has been properly signed. You are now ready to deploy your signed RIA. Page 15 The Trusted-Library attribute is used for applications and applets that are designed to allow untrusted components. No warning dialog is displayed, and an application or applet can load JAR files that contain untrusted classes or resources. Set the value of the attribute to true, for example: Trusted-Library: true This attribute prevents components of a privileged application or applet from being reused with untrusted components. All classes and resources in a JAR file containing this manifest attribute must be signed and request all In a mixed code application or applet, all privileged classes and resources must be included in JAR files that contain the Trusted-Library attribute. This attribute is used for calls between privileged Java code sandbox Java code. If you have JavaScript code calling Java code, see Caller-Allowable-Codebase Attribute. All trusted libraries JARs are loaded in a separate dedicated class loader that is unique to the application or applet instance. This Trusted-Library load device is now the parent of the normal web boot or applet class charger. For backward compatibility with the original search order, both loaders work together to implement a common class path. Consistent with previous releases, JAR files use lazy downloading and are opened as needed to find requested classes and resources. Code in a JAR file to be marked with the Trusted-Library manifest attribute may need to be slightly changed if it uses calls that are class loader dependent, such as the only parameter version of Class.forName(), Class.getResource(), and Class.getResourceAsStream(), some variants of java.util.ResourceBundle.getBundle(), and any other methods that work in relation to their immediate dial-up definer. Changes only need to be made if the requested class or resource can be found in a JAR file that is not a Trusted Library (and therefore loaded by the normal webstart or applet class in loader). Code in a Trusted Library can look up the normal loader by invoking Thread.currentThread().getContextClassLoader(). Note, however, that there are unusual circumstances under which getContextClassLoader() can return null. This can happen, for example, when the garbage collector uses a JRE system thread to invoke the Object.finalize() method for an instance that cannot get to anyone. If you need to convert class to Class.getResource() or Class.getResourceAsStream() to their ClassLoader counterparts, remember to adjust the string parameter as described in the documentation for these two methods. If the original resource name started with '/', then it was an absolute name and the leading '/' simply needs to be removed. Otherwise, determine whether the class instance that was the target of the getResource call is in a named packet. If it is an array, you should first determine the underlying component type for the array. Invoke Class.getName() on the class or component type instance. If the class name contains any '.' characters, it is in a named package that will need to be reconciled to the original resource name. Determine the package name by removing any characters after, and including, the closing character '.'. Next, replace any remaining '.' characters with '/' characters. Finally, add a trailing '/' and add the original resource name string. This new string can now be sent to the ClassLoader version of the getResource() or getResourceAsStream() methods. Generally be sure that the code in the trusted library is written in a carefully and secure manner and is compatible to be loaded in a separate class load instance from any remaining cans included in the program that are loaded by the normal loader. The Permissions attribute and codebase attribute were introduced in the JDK 7u25 edition to defend the RIs against unauthorized code repurchases. Without the Permissions attribute, an attacker may be able to exploit a user by deploying an application that is re-signed with your certificate and running the application at a different permission level. If the Codebase attribute does not specify a secure server, such as HTTPS, there is some risk that your code could be reused in Man-in-the-Middle (MITM) attack schedules. The following code example shows the attributes to be added to the manifest if you have an RIA running in the security sandbox that is expected to be accessed from : Permissions: sandbox Codebase: If RIA is also available from example.backup.com:8080, include both domains for the Codebase attribute: Codebase: example.backup.com:8080 Page 16I is using a third-party Java implementation. Are they affected in the same way? Answer: Contact your supplier for advice on their implementation. I'm a developer. What are the security exceptions that are added with this enhancement? Answer: The following SecurityException messages are described for informational and troubleshooting purposes only. The actual message content may change between implementations and releases. These SecurityExceptions are discarded when a JAR file contains one of the manifest attributes and the JAR file itself contains untrusted components. tried to open sandboxed jar + url+ as Trusted-Only tried to open sandboxed jar + url + as Trusted-Library The following SecurityException is discarded when a JAR file contains the Trusted-Only manifest attribute and untrusted components have previously been accessible. tried to open Trusted-Only jar + url + on sandboxed loader The following SecurityException is discarded when at least one JAR containing the Trusted-Only manifest attribute is opened and a subsequent attempt is made to load an untrusted component. Trusted-Only loader tried to load sandboxed resource from + url The following two SecurityExceptions are discarded when mixed components are first detected and a decision is made not to allow mixing. In the first case, everything previously loaded was trusted and then an attempt was made to load an untrusted component. The second case is the reverse state. trusted loader tried to load sandboxed resource from + url sandboxed loader tried to load trusted resource from + url The following two SecurityExceptions thrown after mixed components had previously been detected and a decision was made to allow them to coexist. The exceptions indicate that a component name collision (resource name or class package name) was detected between trusted and components and the request to load the resource or class was denied. resource \ + name + \ does not match the trust level of other resources of the same name class \ + packageName + \ does not match the level of trust of other classes in the same package The following two SecurityExceptions are discarded when untrusted components have previously been accessed, an attempt to load a trusted component was detected previously, and a decision was made to allow mixed components to coexist, and a JAR containing trusted components is opened and a component name collision is detected between trusted and non-trusted components. untrusted resource \ + name + \ in the path class of untrusted class packages \ + packageName + \ in the path class I have a mixed code Java Web Start application that cannot be easily updated to use the Trusted-Library manifest attribute. Can I sign the JAR files in the sandboxed JNLP without having to change the JNLP to request the security model with all permissions? Answer: Yes, with some limitations. The Sandbox jar files must be signed with the same signing certificate as one or more of the trusted JAR files in a JNLP file that uses the security model with all permissions, and the trusted JAR file must be opened by Java Web Start before any sandboxed resource is loaded that shares the same signatory. This means that the trusted JAR file must be earlier in Java Web Start's JAR search order or it is triggered to load independently of the simple search order using the JAR indexing feature. In Java Web Start, the main application JNLP's JARs are searched first, followed in the declaration order of any JNLP extensions. JAR files labeled within a JNLP that are eagerly searched first, followed by lazy JAR files, followed by any JAR files labeled that use the part feature. I have Java on my phone. Is this affected by this question? Answer: No, Java ME is not affected. Page 17 Example 28-1 allows all RICs from run without security precautions. Ria from other locations does not match the rule, so standard processing is used and the security updr in question is displayed as applicable. To ensure that all RICs are managed by the rule set, you can provide a final rule that matches any RID that was not matched by a previous rule. The action for this rule must be either block or default. Example 28-2 allows all RDs from run without security backups and blocks all other RRs. The default message appears when a RIA is blocked because no custom message is provided. Rules are processed in the order in which they appear in the rule set. Complex patterns can be defined for matching rules by placing the rules in the correct order. Example 28-3 allows RIA from to run without security displays using a secure version of the Java 1.7 platform, but uses standard processing for RIA from which displays applicable RIA from other locations also does not match standard processing is used. Example 28-4 modifies the previous rule set and requires only RIA named Solitaire from run with default processing. Other IRs from are allowed to run without security intervention using a secure version of the Java 1.7 platform. All other RMs are blocked, and a custom message appears. If you want to allow multiple RUs from multiple sites to run, and all RUs are signed with the same certificate, you can use the certificate element to identify the RUs with a rule instead of creating rules for each site and title. Example 28-5 allows any RIA signed with the certificate used by Oracle to run without security support using a secure version of the Java platform. RIA from any host that ends example.com is allowed to run with standard processing. All other RMs are blocked, and a custom message appears. To force the use of a specific JRE, use the power attribute for the action element. This attribute is introduced in the 1.1 version of the deployment rule set. Example 28-6 allows RIA from to run without security dialing using version 1.8_20 of JRE. Any version requested by RIA is ignored. If version 1.8_20 is not available, RIA is blocked. All other RMs are blocked, and a custom message appears. The following example shows the DTD for version 1.1 of the Deployment Rule Set. Version 1.1 is supported by JRE 8u20 and above. Version 1.0 is supported by JRE 7u40 and above. Items introduced after version 1.0 are noted. <! ELEMENT Rule Set (Rule*)> <! ATTLIST Rule Set Version CDATA #REQUIRED> <! ELEMENT rule (id, action)> <! ELEMENT ID (certificate?) > <! ATTLIST id title CDATA #IMPLIED> <! ATTLIST id site CDATA #IMPLIED> <! ELEMENT Certificate TOM> <! ATTLIST Certificate Algorithm CDATA #IMPLIED> <! ATTLIST Certificate Hash CDATA #REQUIRED> <! ELEMENT action (message?) > <! ATTLIST action permission (run | block | default) #REQUIRED> <! ATTLIST Action Version CDATA #IMPLIED> <! ATTLIST Action (true|false) false> <!-- introduced in 1.1 --> <! ELEMENT message (#PCDATA)> <! ATTLIST message locale CDATA #IMPLIED> #IMPLIED>