Frequently Asked Questions - Java

Questions

Answers

1. Questions about Java

1.1. I have a Java-(security/cryptography) problem. Can you help me?

Go to the java forum of Sun. You can find forums where you can ask questions like "How do I generate a keypair", etc.

1.2. I have a Java-XML problem.

Go to the java forum of Sun, section Java Technology & XML and have a look at Apache Xerces.

2. Questions about this package

2.1. I'm using Crimson, but it throws Exceptions. Why?

Crimson is not supported at the moment. The main reason is that Crimson did not support the

`org.w3c.dom.traversal.TreeWalker` interface in the past. Additionally, it does not support the `org.apache.xerces.dom.DocumentImpl.putIdentifier(String ID, Element e)` functionality where Xerces allows us to enable ID attributes during document generation.

Use Apache Xerces instead of Crimson.

2.2. What's up with the Bouncy Castle CSP? / Where is my CSP?

There is *no* JCE bundled together with this distribution. This is because the Apache Project web site is hosted in the US where some export restrictions apply to the cryptographic primitives.

The nice guys from the Legion of Bouncy Castle where so helpful to supply their JCE in a simple JAR package so that we can simply fetch it during the compilation process and put it into the `libs/` directory. When you use the ant makefile `build.xml` and simply say `ant compile` or `ant get-jce`, `ant` tries to fetch this JAR from the australian server. After that step, the compilation works completely.

The ant make tools initiates an automated download of the BouncyCastle JCE. The file is downloaded into the `libs/` directory and a "bc-" is prepended to the filename. This is done because we want the provider name (bc means BouncyCastle) being visible in the JAR's filename.

> **Note:**
> The fact that we *use* Bouncy in this project does not mean that you *must* use it, it's only the default. If you take a look at the configuration `src/org/apache/xml/security/resource/config.xml`, you'll notice the sections which do integrate these alternative JCEs.

More information can be found in the Installation section.

2.3. How do I enable/turn off logging?

The logging is configured in the `config.xml` file which either in the `xmlsec.jar` file or in the class path. This is a little bit complicated as config.xml is used both for library wide configurations like algorithms as well as for the user setting about log4j. This will be changed someday ;-))

OK, so it goes: In the `xml-security/src/org/apache/xml/security/resource/config.xml` file, there is an element called `<log4j:configuration>`. This element contains the XML style configuration information as defined in the log4j DOMConfigurator class . You

can find examples here

2.4. What is the meaning of BaseURI?

When you work with URIs like "`http://www.example.com/index.html`", it is quite sure what you mean as this is an absolute URL, i.e. it is clear which protocol ise used to fetch which file from which server. When you use such a URL inside a signature, the software can automatically figure out what you sign. But when you sign something in your local file system or if you use a relative path like "`../1.txt`", it's not possible to understand this reference without some context. *This* context is the `BaseURI`. For instance, if you sign `URI="../1.txt"` and the `BaseURI="file:///home/user/work/signature.xml"`, it is clear that the file `BaseURI="file:///home/user/1.txt"` is to be signed. But when you create the signature, the file `BaseURI="file:///home/user/work/signature.xml"` does not yet exist; therefore, you have to supply the URL where you intend to store the signature later (relative to the signed objects).

The String BaseURI is the systemID on which the Object will be stored in the future. This is needed to resolve relative links in the `Reference` elements which point to the filesystem or something similar.

Example: Imagine that you want to create a signature to store it on a web server as `http://www.acme.com/signatures/sig1.xml`. So `BaseURI="http://www.acme.com/sig1.xml"`. This means that if you create a `Reference` with `URI="./index.html"`, the library can easily use it's HTTPResourceResolver to fetch `http://www.acme.com/index.html` without that you have to say `URI="http://www.acme.com/index.html"`.

2.5. How do I use the package to generate and verify a signature?

Checkout the samples in `src_samples/org/apache/xml/security/samples/signature/`.

> **Note:**
> The samples divide into two groups: Samples that *create* and samples that *verify* Signatures. Eventually, you should adjust the verifying program to another filename if you get `FileNotFoundException`s.

2.6. I'm using SUN JDK v1.4.0 or v1.4.1 and it get some exceptions. Any clues?

After SUN released the Java (TM) 2 Platform Standard Edition v1.4.0 , the xml-security package stopped working. This is a known problem: SUN packaged a beta of Xalan into the

JDK1.4.0, but the xml-security package requires a stable version of Xalan (v2.2.0 or later). To fix the problem, you have to put the xalan.jar into a special directory in your JDK: `j2sdk1.4.0/jre/lib/endorsed/xalan.jar`. If you installed an out-of-the-box JDK1.4 (e.g. on Windows 2000), the "endorsed" directory does not exist: you'll have to create it by hand.

Note:

Putting this JAR to another location like lib/ext WILL NOT WORK.

For more on that, you can also check the Unofficial JAXP FAQ .

2.7. I get a NullPointerException, and I don't know what's wrong.

Often, this problem is caused by using DOM1 calls like `createElement()`, `setAttribute()`, `createAttribute()`. These are non-namespace-aware and will cause XPath and C14N errors. Always use the DOM2 `create(Attribute|Element)NS(...)` methods instead, even if you're creating an element without a namespace (in that case, you can use null as a namespace).

The Xalan-J Team told us that DOM1 calls are deprecated and are not to be used in code. xml-security has been reviewed and is DOM1 clean now. The Xalan folks told us that if you create Elements or attributes using DOM1 calls which are not namespace aware, they do not care about any problem you have because of incorrect hehaviour of Xalan.

2.8. I sign a document and when I try to verify using the same key, it fails

After you have created the XMLSignature object, before you sign the document, you *must* embed the signature element in the owning document (using a call to `XMLSignature.getElement()` to retrieve the newly created Element node from the signature) before calling the `XMLSignature.sign()` method,

During canonicalisation of the SignedInfo element, the library looks at the parent and ancestor nodes of the Signature element to find any namespaces that the SignedInfo node has inherited. Any that are found are embedded in the canonical form of the SignedInfo. (This is not true when Exclusive Canonicalisation is used, but it is still good practice to insert the element node prior to the sign() method being called).

If you have not embedded the signature node in the document, it will not have any parent or ancestor nodes, so it will not inherit their namespaces. If you then embed it in the document and call `verify()`, the namespaces will be found and the canonical form of SignedInfo will be different to that generated during `sign()`.