JSPservletPkg
Diskless version

TABLE OF CONTENT

TABLE OF FIGURES

1. Foreword

This document describes the diskless version of JSPservletPkg.
JSPservletPkg is a complete implementation for servlets and JSP handling from a remote repository and with dynamic update.

The diskless version is based on the embedded version and like embedded version has been tested so far only with Java Embedded Server 2 (JES 2) from Sun.
The document doesn't refer to the Embedded version.

2. Architecture and principles

Diskless version can be configured either as a diskless client that can ROMed and doesn't need access to a local drive or either as a diskless server that is:

1. A full embedded server able to host archives
2. A server able to store configurations, logs and statistics for any number of diskless clients

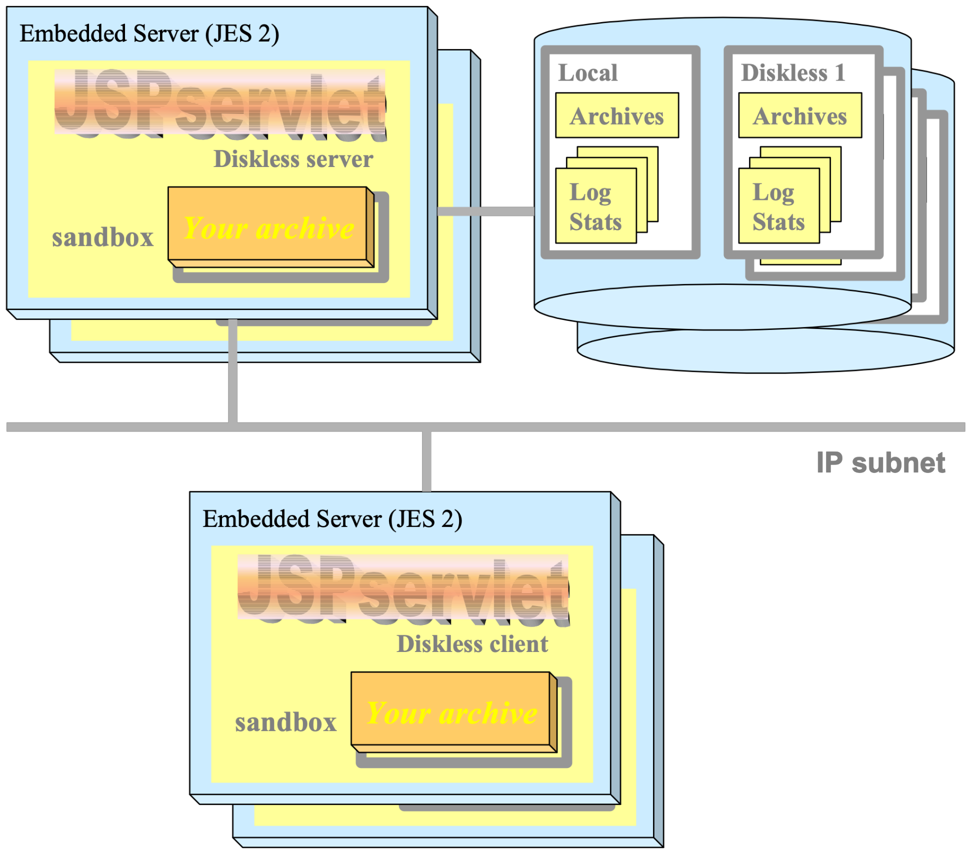


Figure 1: Diskless architecture

Pagebox instances remain independent:

- Each Diskless server maintains its local data (archive list, log and statistic files).
- It also maintains data on behalf of diskless clients.
- Diskless data are replicated. If on a given subnet (in case of multicast), there are 20 diskless clients and 2 diskless servers, each diskless server will maintain the data of the 20 clients

Client data size is small but possibly logs. As diskless clients also send log messages on the wire, it has a consequence. Whereas intensive logging has a negligible impact for Application Server and Embedded server version it should be avoided as much as possible on diskless clients.

Diskless clients and server communicate using new ICP messages, so in UDP either in unicast or multicast.

The protocol is designed for fault tolerance.

3. ICP messages

3.1. Standard

Diskless version uses ICP messages as defined in RFC 2186 and RFC 2187.
They have a header described by the following C structure:

```
struct _icp_common_t {
   unsigned char opcode;          /* opcode */
   unsigned char version;         /* version number */
   unsigned short length;         /* total length (bytes) */
   u_num32 reqnum;                /* req number (req'd for UDP) */
   u_num32 flags;
   u_num32 pad;
   u_num32 shostid;               /* sender host id */
   /* diskless extension */
   u_num64 lastModified;          /* timestamp */
   char[] ID;                     /* issuer unique identifier
                                   *  variable length delimited by a 0 and followed by the payload */
};
```

The opcode describes the message. Diskless version uses opcodes undefined in the standard.
It doesn't use flags and pad fields and sets the version to two.

3.2. Messages

3.2.1. ICP_OP_INI

Opcode: 5.
PageBox initialization message issued by diskless client.
Payload: none.
Purpose: Query for archive names.

3.2.2. ICP_OP_ARCHIVE_STATUS

Opcode: 6.
PageBox response returned by diskless server when it receives an ICP_OP_INI.
Payload: serialized properties whose keys are archive names and values are download locations.
Purpose: Provide the list of archives to manage to the diskless clients.

3.2.3. ICP_OP_ARCHIVE_UPDATE

Opcode: 13.
PageBox archive update message sent by diskless client.
Payload: serialized properties whose keys are archive names and values are download locations.
Purpose: notify diskless servers updated list of archives managed by the client.

This message is sent periodically (every SCANPERIOD = 1 minute in the current version) to address cases where a diskless server is not up when the diskless client adds, delete or change an archive.

3.2.4. ICP_OP_STAT_QUERY

Opcode: 7.

Query for statistics issued by diskless client.
Payload: none. The ID is made of the unique identifier of the client followed by slash (“/”) and by the archive name.

The initialization protocol is:

Client		**Server**
ICP_OP_INI	==>	
	<==	ICP_OP_ARCHIVE_STATUS
For each archive managed by the client		
ICP_OP_STAT_QUERY	==>	
	<==	ICP_OP_STAT_STATUS

3.2.5. ICP_OP_STAT_STATUS
Opcode: 12.
PageBox response returned by diskless server when it receives an ICP_OP_STAT_QUERY.
Payload: serialized properties whose keys are servlet names and values are invocation numbers.
Purpose: Provide current statistics to the client before first invocation by browser users.

3.2.6. ICP_OP_STAT _UPDATE
Opcode: 14.
PageBox statistics update message sent by diskless client.
Payload: serialized properties whose keys are servlet names and values are invocation numbers.
Purpose: Save statistics on server side.

This message is sent periodically (every SCANPERIOD = 1 minute in the current version).

3.2.7. ICP_OP_LOG
Opcode: 8.
PageBox log message sent by diskless client.
Payload: message sent by diskless client.
Purpose: Update log on server side.

3.2.8. ICP_OP_FIN
Opcode: 9.
Message issued by diskless client to notify diskless servers of its termination.
Payload: none.

When a diskless server receives ICP_OP_FIN, it closes the relevant log file (identified by the ID carried in ICP_OP_FIN.

3.3. Protocol
The protocol is designed for fault tolerance. The unavailability of a server at a given time doesn’t compromise the operations of a set of diskless servers and clients.

3.3.1. Poll mechanism
At initialization, a diskless client sends an ICP_OP_INI message to all servers.
All up servers answer and the client selects the ICP_OP_ARCHIVE_STATUS answer whose lastModified field is the most recent to get the most up to date archive list. If a server was not up when the client updated lastly its archive definition, its response is not used.

The same mechanism is used for statistics (messages ICP_OP_STAT_QUERY and ICP_OP_STAT_STATUS).

3.3.2. Periodic refresh
Suppose the administrator adds an archive to the archive list of a diskless client.

Then the client sends an ICP_OP_ARCHIVE_UPDATE message. Server A is up and records the update. Then it is stopped. A server B is started.
Now the administrator changes the location of an archive of a diskless client.
The client sends an ICP_OP_ARCHIVE_UPDATE. Server B records that update.
Now neither server A nor server B has an information that describes correctly the state of the client.

To address this issue, update messages are send periodically – even if no update occurs in a given session in order to refresh servers with a longer inactivity. It has also an important consequence. One can add a diskless server dynamically. One minute later, it will be up to date.

4. Parameters

This version is designed to be deployed in JES2 bundles. So its archive must have a Manifest like this:

```
Bundle-Name: JSPservlet
Bundle-Description: JSPservlet
Bundle-Vendor: Alexis Grandemange (GNU GPL 2).
Bundle-Version: 1.0.1
Bundle-DocURL: http://java.sun.com/products/embeddedserver
Bundle-ContactAddress: alexis.grandemange@amadeus.net
Bundle-Activator: JSPservletPkg.JSPhandler
Import-Package: org.osgi.service.http,
  javax.servlet; specification-version=2.1.1,
  javax.servlet.http; specification-version=2.1.1,
  com.sun.jes.service.http.auth.basic
Import-Service: com.sun.jes.service.http.auth.basic.BasicSchemeHandler
```

There is no limitation to the number of bundles, which can be configured, in a given Application Server.

Its configuration is specified by a property file named *Bundle-Name*.properties where *Bundle-Name* is the name you gave to the bundle in the manifest. Here is an example of configuration:

```
cachePath=/jdj
toTrace=TRUE
toStat=TRUE
logfile=/jdj/log.txt
remoteLocations=/jdj/jdj.properties
allPermissionPolicy=C:/temp/JES2/allPermission.policy
defaultPolicy=C:/temp/JES2/default.policy
keystore=C:/temp/JES2/keystore
keystorePassword=keystorePswd
adminUser=admin
adminPasswd=admin
ID=Hamlet
contextPath=/jdj
useBundleSpace=TRUE
Diskless=FALSE
MulticastGroup=228.10.10.10
MulticastPort=9089
Multicast=TRUE
CRLURL=ldap://localhost/CN=alexis,CN=agrandem,CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC=alexis
CAURL=ldap://localhost/CN=alexis,CN=AIA,CN=Public Key
Services,CN=Services,CN=Configuration,DC=alexis
CRLLDAPuser=Alexis/Users/Administrator
```

```
CRLLDAPpasswd=XXXXXX
CALDAPuser=Alexis/Users/Administrator
CALDAPpasswd=XXXXXX
expiration=30
CRLperiod=30
```

This file acts as the web.xml of Application Servers' version.
I bolded the parameters that doesn't exist in web.xml version.
The package must contain:

```
WEB-INF
   web.xml  // as the file above
   classes
       JSPupdate.class
       JSPservletPkg
          CRLchecker.class
          DisklessLog.class
          ICPhandler$DisklessCltHandler.class
          ICPhandler$IcpHeader.class
          ICPhandler$Log.class
          ICPhandler$LogPrivilegedAction.class
          ICPhandler.class
          JSPhandler$ClassEntry$ServletInfo.class
          JSPhandler$ClassEntry$Stat.class
          JSPhandler$ClassEntry.class
          JSPhandler$Log.class
          JSPhandler$Scanner.class
          JSPhandler.class
          JSPloader$ClassInfo.class
          JSPloader$ProtectionDomainInfo.class
          JSPloader$ResourceEntry.class
          JSPloader.class
          JSPloaderException.class
          JSPresourceServlet.class
          ResourcePrivilegedAction.class
          PageBoxAPI.class
          ServletLog.class
          ServletStat.class
          ServletUpdate.class
```

We detail below JSPservlet.properties parameters.

4.1. cachePath

cachePath is the location where jars are locally stored after been retrieved from remote location.
Default value: C:/temp.

4.2. toTrace

Tells if the tool must write diagnostic messages.
Default value: false.

4.3. toStat

Tells if the tool must record statistics.
Default value: false.
If true, statistics are recorded **per archive** in a file *cachePath*/*archive*.stat, which is a property file, for instance:

```
#Sun Nov 19 23:20:32 CET 2000
TestServlet/FileAccess=3
helloImg=8
TestServlet/OtherServlet=2
SnoopJSP=3
TestServlet/ForwardingServlet=4
SnoopServlet=2
```

4.4. ID

Allows the deployer to specify a unique identifier.
PageBoxAPI allows retrieving this ID.
Application: deployment of a large number of instances.

No default value.

4.5. logfile

Tells where the tool must write diagnostic messages.
Default value: $CachePath/log.txt.

4.6. remoteLocations

Location of a property file containing jar names and associated URLs.
Default value: $CachePath/$ContextPath.properties where ContextPath indicated the name the war file is deployed with.

JSPupdate updates this file. It is OK to modify it manually but don't expect to retrieve your comments.

4.7. expiration

You can set this parameter to minimize the round trip number between the browser and the server.
JSPservlet sets the *Expire* header field of static content (content with an extension different of *class*). It computes the *Expire* as current_time + *expiration*.
expiration unit is second.

Default value: 5 seconds.

4.8. contextPath

contextPath is the equivalent of the name you give to the JSPservletPkg Web Archive in Application Servers configuration.
Consider you configure JES2 http server to handle requests toward http://myserver:8080, you will invoke a servlet defined in archive myarchive.jar in mypath/myservlet with http://myserver:8080/contextPath/myarchive/mypath/myservlet.

Default: none. Must be specified.

4.9. useBundleSpace

Boolean. If true, uses JES 2 BundleContext's getDataFile to create File objects in the persistent storage area provided for the bundle by the framework.
I recommend setting it to true first because it is a requirement for well-behaving bundles and second because it provides useful features such as removing all bundle files when the bundle is uninstalled.

If you set it to true, path parameters are relative to the bundle space root but:
- allPermissionPolicy

- defaultPolicy

The reason is these files must be installed separately and are not related to a bundle.

Default: true.

4.10. Diskless

Boolean. If true, starts as a diskless client and doesn't write on local file system. In this case, you don't need to set parameters such as logfile and cachePath. If false, starts as a diskless server and writes on local file system for it and its clients.

Default: false.

4.11. Multicast

Boolean. If true, uses Multicast. In this case, it uses the multicast group specified by MulticastGroup and the port specified by MulticastPort, both for listening and sending.

If Multicast=false, uses Unicast. In this case, listens on the address specified by MulticastGroup and the port specified by MulticastPort. A diskless client must be configured with two other parameters:

❑ RemAddrs contains the IP addresses of diskless servers
❑ RemPorts contains the ports of diskless servers

Default: true.

4.12. MulticastGroup

Specify a valid multicast group with a string xxx.xxx.xxx.xxx, for instance 228.10.10.10 if Multicast=true. Otherwise specifies an IP address or name.

Default: none. Mandatory.

4.13. MulticastPort

Specify an IP port.

Default: none. Mandatory.

4.14. RemAddrs

Contains the IP addresses of diskless servers as strings separated by commas, for instance Miami, Orlando, Lauderdale.

4.15. RemPorts

Contains the IP ports of diskless servers as strings separated by commas, for instance 9001, 9000, 9002.
In this example, the diskless client will send messages to Miami:9001, Orlando:9000, Lauderdale: 9002.

4.16. allPermissionPolicy

If it is set, *allPermissionPolicy* is the path to a policy file with syntax conforming to the Java 2 security specification. If *defaultPolicy* is also set it means:

1. JSPservletPkg will implement sandboxes. So every archive will run with the permissions defined either in *cachePath*/*archive*.policy or in *cachePath*/java.policy where:
    - cachePath is the cachePath initialization parameter value
    - archive is the archive name without suffix
2. The Java server itself will run with the permission described in *allPermissionPolicy*.

The following no-brainer *allPermissionPolicy* will work in all cases:

```
grant {
        permission java.security.AllPermission;
};
```

Note that you should never grant permissions in *cachePath/archive* policy as these files are downloaded from the archive location.

Default: the parameter has no default value.

4.17. defaultPolicy

If it is set, *defaultPolicy* is the path to a policy file with syntax conforming to the Java 2 security specification. If *allPermissionPolicy* is also set it means:

1. JSPservletPkg will implement sandboxes. So every archive will run with the permissions defined either in *cachePath*/*archive*.policy or in *cachePath*/java.policy where:
   - cachePath is the cachePath initialization parameter value
   - archive is the archive name without suffix

   If no policy applies to the archive, then *defaultPolicy* is used
2. The Java server itself will run with the permission described in *allPermissionPolicy*.

The Java server itself will run with the permission described in *allPermissionPolicy*.

Default: the parameter has no default value.

4.18. keystore

keystore is the name of the key store in cachePath directory, for instance "keystore" but not "/mydir/keystore" or "mydir/keystore".

If *keystore* is set, when JSPservletPkg downloads an archive, it tries

- To download a certificate from the same location as the archive and named *archive*.cer. If it finds, it adds the certificate to *keystore*, which has to be in Sun JKS format with an *archive* alias
- To download a permission file from the same location as the archive and named *archive*.policy. This file should only contain permission entries. JSPservletPkg adds a keystore line, builds the appropriate grant line and stores it in *cachePath*/*archive*.policy in order to implement a sandbox with the permissions requested by the provider.

The archive user has no longer to administrate security. It is appropriate for trusted providers.

Default: the parameter has no default value.

4.19. keystorePassword

Password JSPservletPkg uses to access the keystore.

Default: the parameter has no default value.

4.20. CAURL

When JSPservletPkg implements sandboxes, it processes signed archives and retrieve classes certificate chain. If CAURL is set, it connects to this URL and expects to retrieve a Certificate Authority certificate used in the classes certificate chain.

If it fails to connect to the CA or if what it retrieves is not a certificate, it logs an ERROR entry with "directory access failure". If the certificate is valid but not present in a class certificate chain, it invalidates the class just as if one of its certificates was revoked.

Note that JSPservletPkg doesn’t load the class and therefore doesn’t raise a security exception but a class not found exception.

If you set this parameter, you MUST add JNDI to your Java Server CLASSPATH in JDK 1.2. In JDK 1.3, you don’t have to, as JNDI is included in JDK.

Default: the parameter has no default value.

4.21. CRLURL

When JSPservletPkg implements sandboxes, it processes signed archives and retrieve classes certificates.
If CRLURL is set, it connects to this URL and expects to retrieve a Certificate Revocation List (CRL) used to check if a class certificate is revoked.

If it fails to connect to the CA or if what it retrieves is not a CRL, it logs an ERROR entry with “directory access failure”. If the CRL is valid and it finds one of the class certificates in it, it invalidates the class.

Note JSPservletPkg doesn’t load the class and therefore doesn’t raise a security exception but a class not found exception.

If you set this parameter, you MUST add JNDI to your Java Server CLASSPATH in JDK 1.2. In JDK 1.3, you don’t have to, as JNDI is included in JDK.

Default: the parameter has no default value.

4.22. CALDAPuser

Principal used to connect to the Directory server to retrieve CAURL.

Default: the parameter has no default value. If it is not set, the tool connects to the Directory server without credential and password (LDAPpasswd).

4.23. CALDAPpasswd

Password used to connect to the Directory server to retrieve CAURL.

Default: the parameter has no default value.

4.24. CRLLDAPuser

Principal used to connect to the Directory server to retrieve CRLURL.

Default: the parameter has no default value. If it is not set, the tool connects to the Directory server without credential and password (LDAPpasswd).

4.25. CRLLDAPpasswd

Password used to connect to the Directory server to retrieve CRLURL.

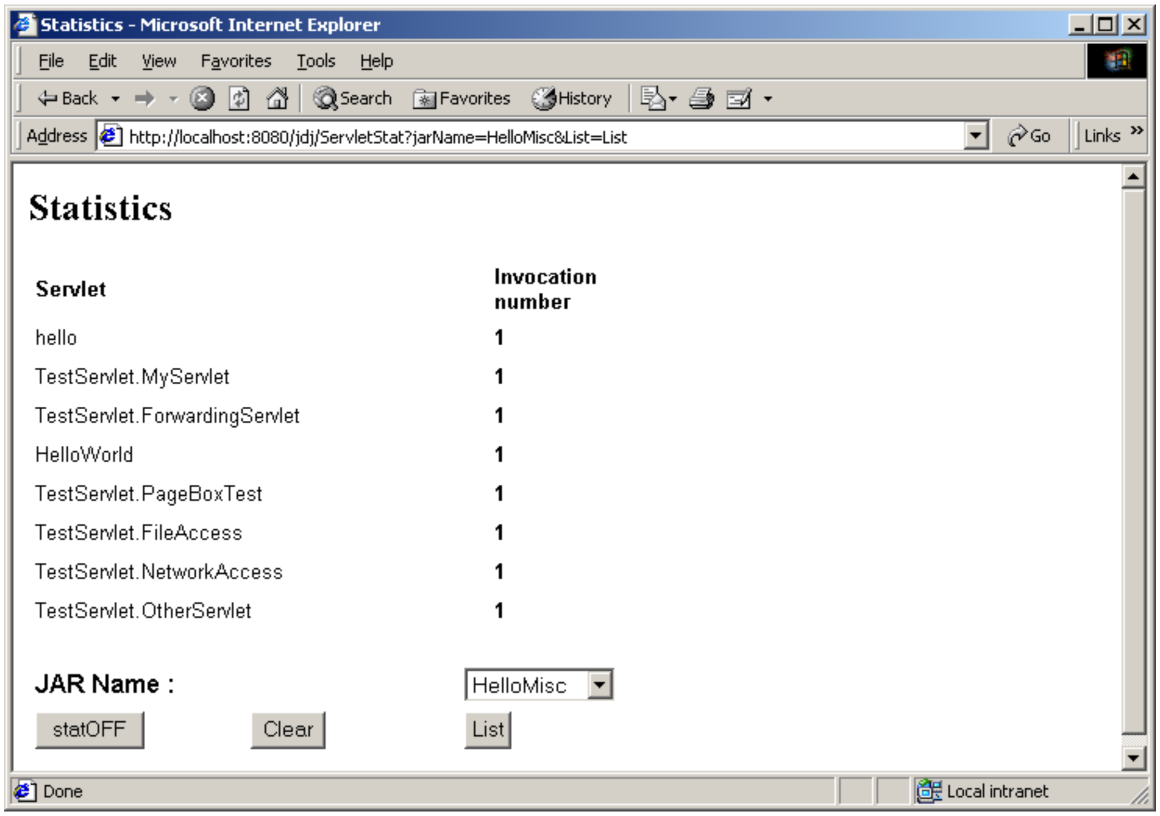
Default: the parameter has no default value.

4.26. CRLperiod

Defines how often the tool will connect to check for CRL updates in seconds.

Default: 7 * 24 * 3600 (1 week).

5. Statistics and troubleshooting


Statistics - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/jdj/ServletStat?jarName=HelloMisc&List=List
Go Links

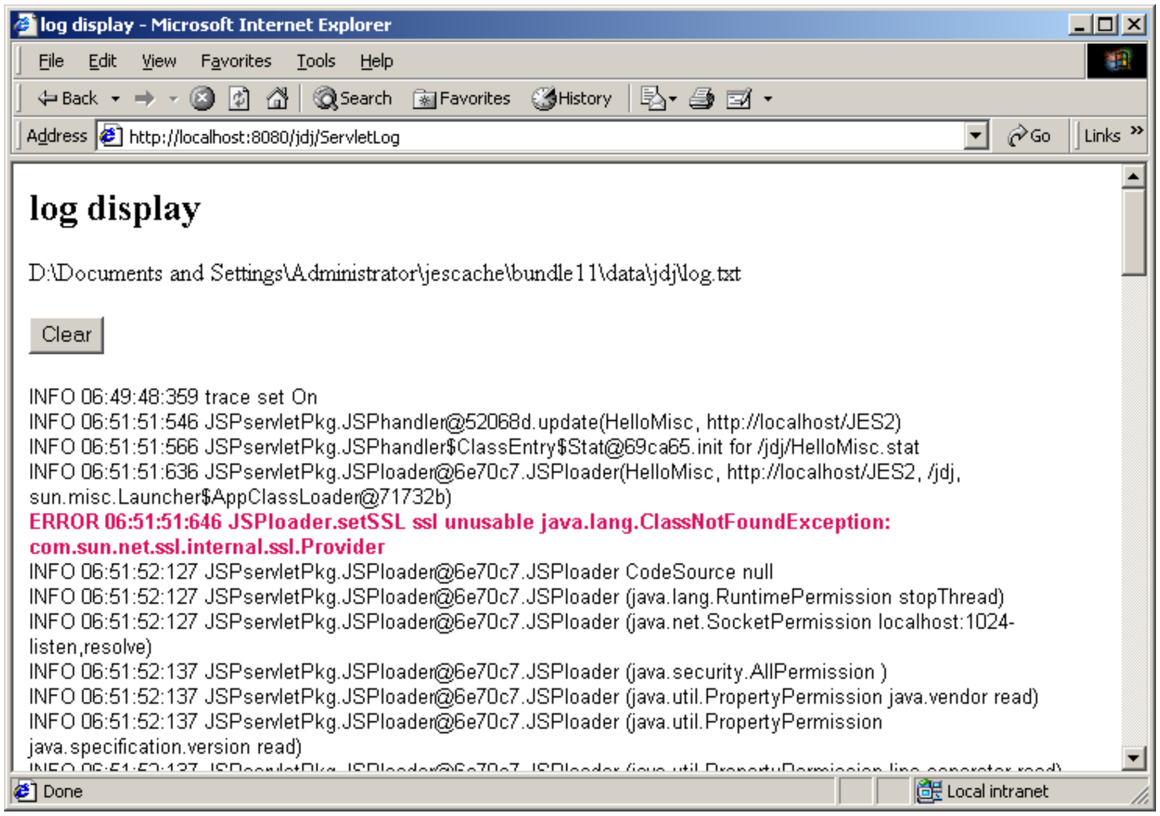
Statistics

Servlet	Invocation number
hello	1
TestServlet.MyServlet	1
TestServlet.ForwardingServlet	1
HelloWorld	1
TestServlet.PageBoxTest	1
TestServlet.FileAccess	1
TestServlet.NetworkAccess	1
TestServlet.OtherServlet	1

JAR Name : HelloMisc

statOFF
Clear
List
Done
Local intranet

Figure 2: Statistics using ServletStat


log display - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/jdj/ServletLog
Go Links
Clear
Done
Local intranet

log display

D:\Documents and Settings\Administrator\jescache\bundle11\data\jdj\log.txt

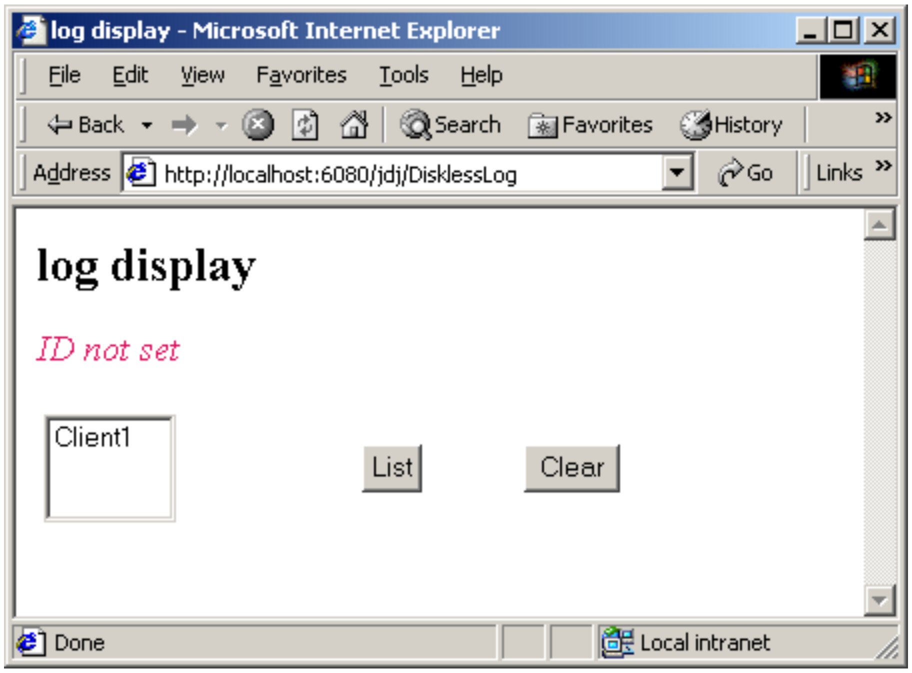
```
INFO 06:49:48:359 trace set On
INFO 06:51:51:546 JSPservletPkg.JSPhandler@52068d.update(HelloMisc, http://localhost/JES2)
INFO 06:51:51:566 JSPservletPkg.JSPhandler$ClassEntry$Stat@69ca65.init for /jdj/HelloMisc.stat
INFO 06:51:51:636 JSPservletPkg.JSPloader@6e70c7.JSPloader(HelloMisc, http://localhost/JES2, /jdj,
sun.misc.Launcher$AppClassLoader@71732b)
ERROR 06:51:51:646 JSPloader.setSSL ssl unusable java.lang.ClassNotFoundException:
com.sun.net.ssl.internal.ssl.Provider
INFO 06:51:52:127 JSPservletPkg.JSPloader@6e70c7.JSPloader CodeSource null
INFO 06:51:52:127 JSPservletPkg.JSPloader@6e70c7.JSPloader (java.lang.RuntimePermission stopThread)
INFO 06:51:52:127 JSPservletPkg.JSPloader@6e70c7.JSPloader (java.net.SocketPermission localhost:1024-
listen,resolve)
INFO 06:51:52:137 JSPservletPkg.JSPloader@6e70c7.JSPloader (java.security.AllPermission )
INFO 06:51:52:137 JSPservletPkg.JSPloader@6e70c7.JSPloader (java.util.PropertyPermission java.vendor read)
INFO 06:51:52:137 JSPservletPkg.JSPloader@6e70c7.JSPloader (java.util.PropertyPermission
java.specification.version read)
```

Figure 3: log using ServletLog

You can use the log path I list at the top of the screen to find out the bundle location. However it is not a hidden secret: It starts with the value of com.sun.jes.framework.bundles.baseurl, which is D:\Documents and Settings\Administrator\jescache in this case, followed by bundle*bundle_id*\data, where *bundle_id* is a number set by the framework.

ServletLog reads the log file. Log entries not being in memory, it can be used only for diskless servers. To display diskless client logs, you must use DisklessLog and use the URL http://serverURL/DisklessLog. Note that if you have two or more diskless servers on the same subnet, you can choose any of them.

Then you are prompted to select the appropriate diskless client:

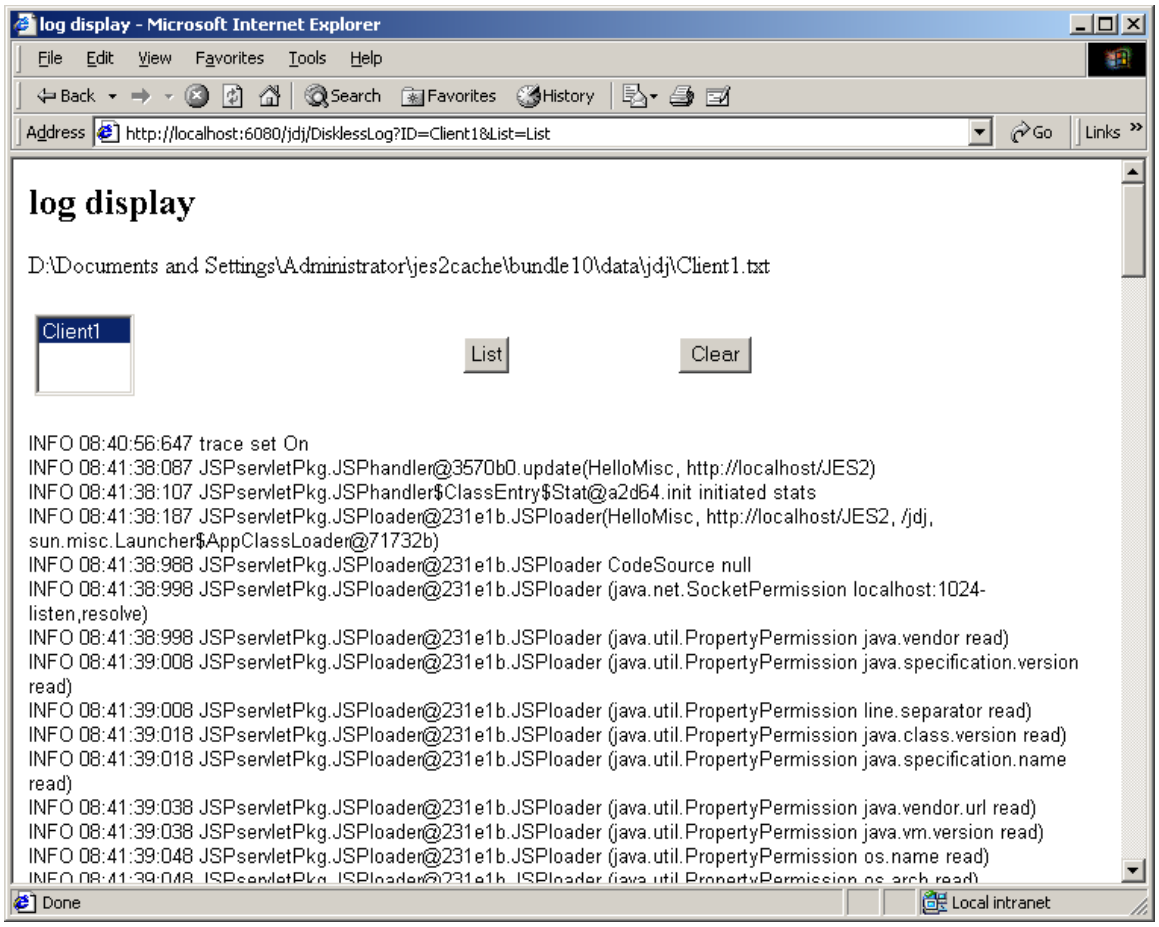

log display - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:6080/jdj/DisklessLog
Go Links
Done
Local intranet

log display

ID not set

Figure 4: Selection of the client on DisklessLog

You select the client and click on List. Then you get the same display as with ServletLog:


log display - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:6080/jdj/DisklessLog?ID=Client1&List=List
Go Links
Client1
List
Clear
Done
Local intranet

log display

D:\Documents and Settings\Administrator\jes2cache\bundle10\data\jdj\Client1.txt

```
INFO 08:40:56:647 trace set On
INFO 08:41:38:087 JSPservletPkg.JSPhandler@3570b0.update(HelloMisc, http://localhost/JES2)
INFO 08:41:38:107 JSPservletPkg.JSPhandler$ClassEntry$Stat@a2d64.init initiated stats
INFO 08:41:38:187 JSPservletPkg.JSPloader@231e1b.JSPloader(HelloMisc, http://localhost/JES2, /jdj,
sun.misc.Launcher$AppClassLoader@71732b)
INFO 08:41:38:988 JSPservletPkg.JSPloader@231e1b.JSPloader CodeSource null
INFO 08:41:38:998 JSPservletPkg.JSPloader@231e1b.JSPloader (java.net.SocketPermission localhost:1024-
listen,resolve)
INFO 08:41:38:998 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.vendor read)
INFO 08:41:39:008 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.specification.version
read)
INFO 08:41:39:008 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission line.separator read)
INFO 08:41:39:018 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.class.version read)
INFO 08:41:39:018 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.specification.name
read)
INFO 08:41:39:038 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.vendor.url read)
INFO 08:41:39:038 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission java.vm.version read)
INFO 08:41:39:048 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission os.name read)
INFO 08:41:39:048 JSPservletPkg.JSPloader@231e1b.JSPloader (java.util.PropertyPermission os.arch read)
```

Figure 5: Diskless client log display with DisklessLog

You see the path of the log and clear it.

6. Notes on implementation

6.1. Dynamic update

Dynamic update of jar file is supported though ServletUpdate.
ServletUpdate display is split in two sections:

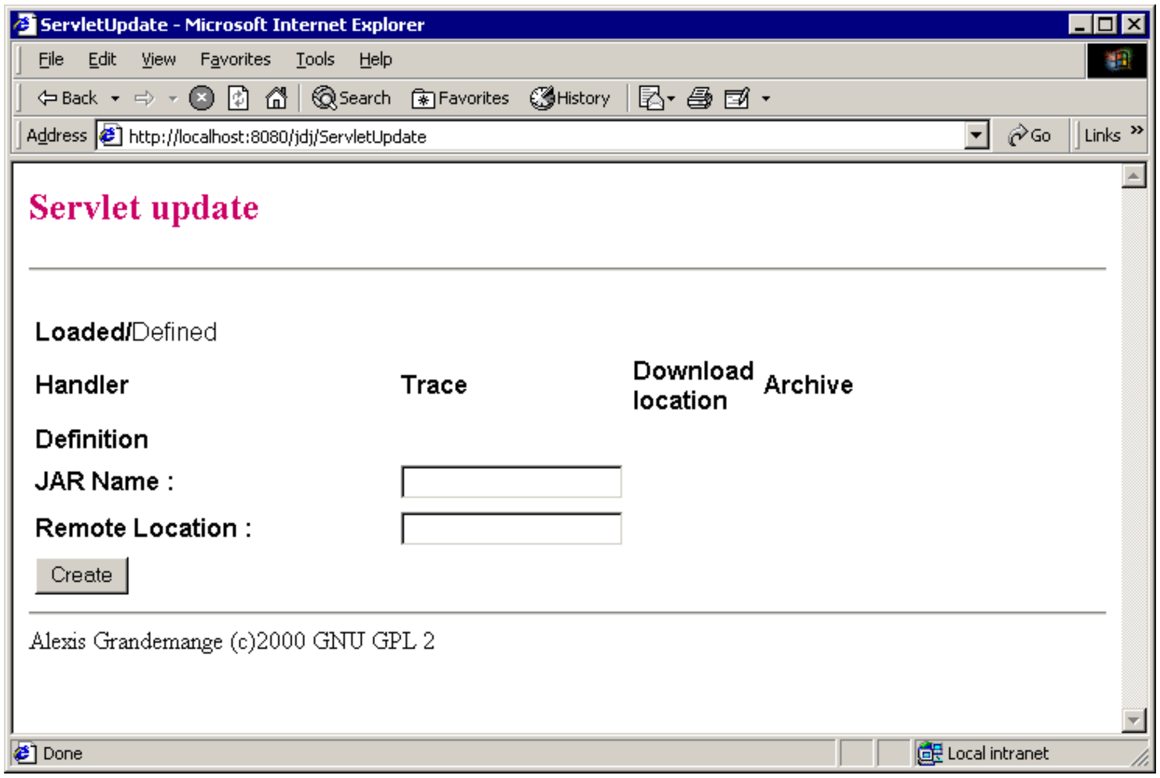
- A *Definition* section at the bottom
- A *loaded/defined* section at the top, that lists known and loaded archives

ServletUpdate uses GET mode. It is a servlet defined in the JSPservlet package.

6.1.1. Definition section

You must fill the JAR Name field without extension. If you don't fill the remote location the current location is reused. It is the JAR file URL minus the file name.
Assuming you specified a jar name myjar and a remote location http://www.mydownloadsite.com, JSPserletPkg will download http://www.mydownloadsite.com/myjar.jar and persist your action in RemoteLocations with a property myjar=http://www.mydownloadsite.com.

Note the URL of ServletUpdate http://localhost:6080/jdj/JSPupdate. jdj is the contextPath. If you deploy different JSPservlet bundles, you must select the appropriate JSPservlet to require an update. When you updates, the local cache is removed and the jar is always loaded from the remote location. You can also use ServletUpdate to add a jar.


ServletUpdate - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/jdj/ServletUpdate
Go Links
Done
Local intranet

Servlet update

Loaded/Defined

Handler	Trace	Download location	Archive

Definition

JAR Name :

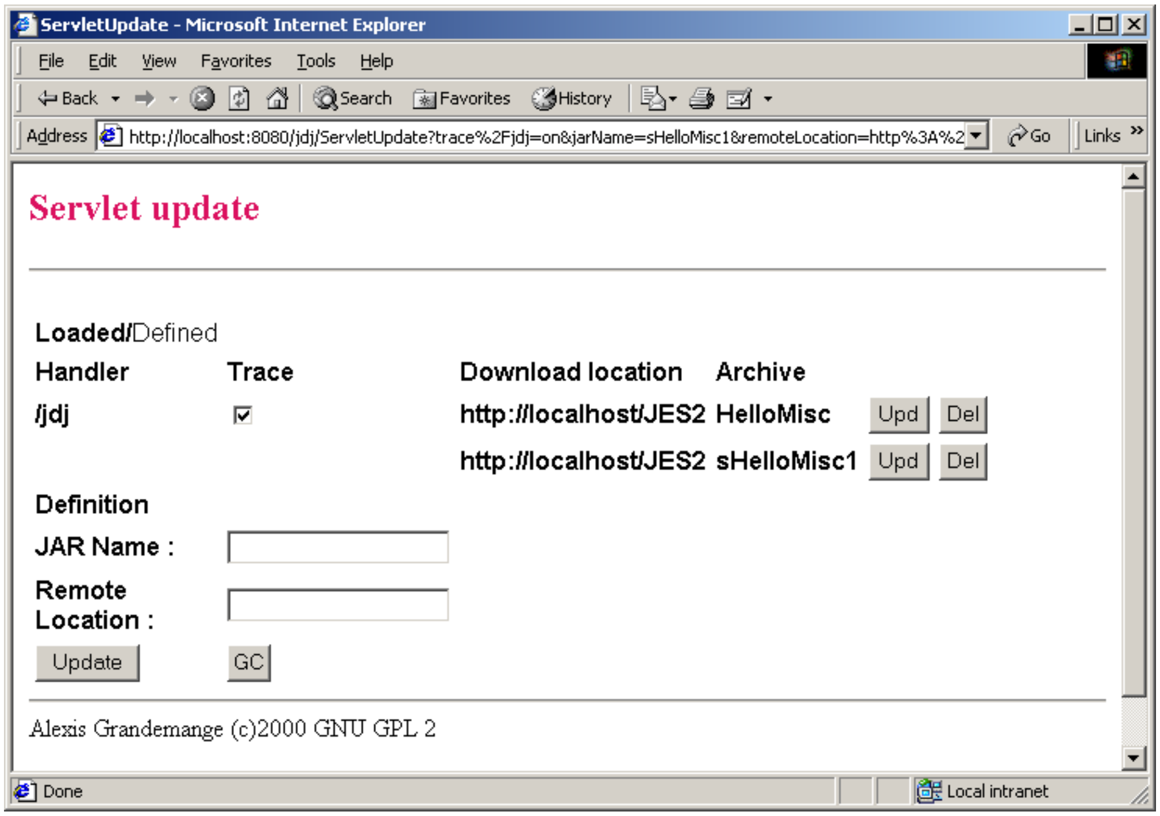
Remote Location :

Create

Alexis Grandemange (c)2000 GNU GPL 2

Figure 6: ServletUpdate definition

6.1.2. Loaded/defined section


ServletUpdate - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/jdj/ServletUpdate?trace%2Fjdj=on&jarName=sHelloMisc1&remoteLocation=http%3A%2
Go Links
Done
Local intranet

Servlet update

Loaded/Defined

Handler	Trace	Download location	Archive		
/jdj	✓	http://localhost/JES2	HelloMisc	Upd	Del
		http://localhost/JES2	sHelloMisc1	Upd	Del

Definition

JAR Name :

Remote Location :

Update GC

Alexis Grandemange (c)2000 GNU GPL 2

Figure 7: ServletUpdate list

It supports a trace checkbox and two buttons:

- If trace is checked, it means that trace is on.
- Upd in the Loaded/define section means
  - Download a new version of the archive in the cache
  - Replace current loaded version of classes and resources
- Del in the Loaded/define section means
  - Remove the archive from the cache
  - Remove the archive servlets, classes and resources from memory

You can safely require an update when pages are accessed.
GC allows to invoke System.gc() in the Java server context.

6.2. Resource handling

Typical case of that is image handling.
Suppose a JSP whose URL is http://www.mysite.com/jdj/myjar/myJSP references an image with a relative path images/myimage.gif. The Application Server looks for http://www.mysite.com/jdj/myjar/images/myimage.gif and therefore invokes JSPresourceServlet.

JSPresourceServlet looks first for images/myimage.gif in the archive. If it doesn't find it, it retrieves the download location in the myjar's RemoteLocations property. If it finds its remote location is www.mydownloadsite.com, it downloads the image from www.mydownloadsite.com/images/myimage.gif.

7. Special cases

7.1. *RequestDispatcher*

The only case where you must modify your code is when you need using a RequestDispatcher either to include or forward a request. You can use JSPservlet.getJAR() helper function.

```
RequestDispatcher rd = getServletContext().getRequestDispatcher(
    JSPservlet.getJAR(getClass().getClassLoader()) + OtherServlet);
```

The reason is as a servlet developer you should not hardcode neither the Web Application name as the standard enforces it, neither the jar name where your servlet will be deployed.

7.2. *Use of SSL*

JSPservlet allows using SSL to download archives using the ssl.jar bundle.

The explanation below is not JSPservlet related but it still can help you to configure the thing. As it can fail at your first attempt, I recommend you reading the JSSE API user guide. It explains you how to trace the SSL connection with javax.net.debug=all and gives you a lot of information.

You need first to SSL enable the Web Server where you download the archive from. It implies generating a key pair. The server keeps the private key internally and asks you to submit a certificate request to the Certificate Authority of your choice.

During the SSL handshaking, jsse will check it knows the certificate chain the Web Server presents it using the keystore specified by com.sun.jes.impl.keystore.store. A default keystore is defined in JES installation path/lib/tlscerts.

tlscerts format is jks, which means you can use keytool to display or update its content.
If the server Certificate Authority certificate is not in the list, you have to add it.
Retrieving it depends on the browser and on the Certificate Authority. You can ask for a DER encoded or for a base 64 format.

Then you can add your server certificate with:

```
keytool -import -file C:\TEMP \serverCA.cer -alias myCA –keystore
<JES2-home>/lib/tlscerts -storepass passphrase
```

7.3. *Load test*

With up to 3000 servlets in a single jar file.
The performance impact of tracing is below 10%.
The sandbox performance impact is not measurable.

7.4. *Functional tests*

7.4.1. Update

ServletUpdate
Multiple archives, multiple web applications
Invalid archive name/location

7.4.2. Security

Non-signed archive

- ❑ without sandbox
- ❑ with sandbox and no definition in policy file
- ❑ with sandbox and permissions granted in policy file.

Signed archive with and without sandbox. With sandbox:

- ❑ Valid certificate

- ❑ Certificate not found in CA
- ❑ Revoked certificate
- ❑ Uncheckable archive
  - CAURL not set or not accessible
  - CRLURL not set or not accessible
- ❑ Revocation when the server is running

1 policy file per web application or 1 per archive.
No permission granted to a signed archive.
Policy/certificate download

7.4.3. Misc

- ❑ Image handling
- ❑ Beans handling
- ❑ Servlet inheritance
- ❑ Servlet/html include and forward
- ❑ Caching
- ❑ JSPservletPkg defined in CLASSPATH
- ❑ https for download

8. Limitations

I checked the tool supports:
- Servlet inheritance, case where you define a servlet as extending a base servlet.
- Tag libs
- JSP beans. Note however a bean is created with Beans.instantiate(). This method tries to restore the bean from a bean.ser resource and if it doesn't find it, it creates it using newInstance(). The tool searches the resource first in the archive, next in the same remote directory as the archive and finally asks the resource to the application server.

Limitations can be:
1. Though it supports the Tomcat jspc servlet compiler, JES 2 doesn't support taglib. If you need it you have to add it yourself
2. HttpJspBase.getClassLoader() doesn't return the JSPservlet class loader that knows how to find bean classes and serialized files

I chose using Sun packages as much as possible.
1. I used the tcatjspcruntime bundle to provide basic JSP support.
2. I created a tagext bundle to add tag support. The purpose of this package is to export javax.servlet.jsp.tagext package that I extracted from Tomcat's servlet.jar. I defined a minimal bundle activator:

```
public class TagextActivator implements BundleActivator {
  public void start(BundleContext bundlecontext) {}
  public void stop(BundleContext bundlecontext) {}
  public TagextActivator() {}
}
```

I packaged the bundle with this manifest:

```
Manifest-Version: 1.0
Bundle-Vendor: apache.org
Bundle-Version: 1.0
Bundle-Activator: tagext.TagextActivator
Bundle-DocURL: http://jakarta.apache.org
Created-By: 1.2.2 (Sun Microsystems Inc.)
Bundle-Name: tagext
Bundle-ContactAddress: agrandemange@amadeus.net
Export-Package: javax.servlet.jsp.tagext
Bundle-Description: tag extension
Import-Package:
javax.servlet,javax.servlet.jsp,javax.servlet.http,org.apache.jasper,org.apache.jasper.runtime
```

3. It is not enough because taglibs have exceptions, JspTagException that inherits from JspException, defined in javax.servlet.jsp but not in tcatjspcruntime. I chose adding them to JSPservlet package.
4. I replaced Beans.instantiate(getClassLoader(), myBean) by Beans.instantiate(getClass().getClassLoader(), myBean) in compiled JSPs.

You can consider an alternate solution where you add the JSP related stuff to JSPservlet.

JSP packaging of JES 2 will probably be enhanced in the coming months and the most important point here is that embedded servers can support JSP V 1.1.

9. Differences between Application Server, JES 2 and diskless versions

Application Server version supports Servlet 2.2.
JES 2 version supports JES 2 Servlet 2.1.
Diskless version is a superset of JES 2 support with a support for diskless – ROM able – clients.
All versions should follow closely framework versions in the future.

The major design difference is between JES/diskless and Application Server version:
- ❑ In the application server version, a service servlet, JSPservlet forwards requests to the target servlet
- ❑ In the JES2 version, the package registers every servlet in the framework. The service servlet, JSPresourceServlet handles only resources.
- ❑ As a consequence, JES 2 framework invokes directly the target servlets.

This has two important implications:
1. In case of archive update, there is no window where a request should hang but there is a window where a request can fail because the previous servlet version was unregistered and the new version was not yet registered.
2. If an archive certificate is revoked, you need to update the archive or restart the server for the revocation to be detected.

I chose to design the JES 2 version in that way to minimize its footprint.

Diskless version is close to JES 2 version because differences are hidden in
- ❑ Initialization
- ❑ Logging method implementation
- ❑ Periodic checks

This code makes however a difference in term of footprint – 40% bigger. I maintain JES 2 version mainly for that reason.

10. Miscellaneous

10.1. License

10.2. Deliveries

Package name: JSPLoaderPkg.

Source files:

CRLchecker.java	JNDI/LDAP code
DisklessLog.java	Servlet to display diskless client logs on diskless server side
ICPhandler.java	ICP message handler
JSPloader.java	class loader
JSPloaderException.java	package exception
JSPhandler.java	Web application handler
JSPresourceServlet.java	Resource handling servlet
PageBoxAPI.java	API to get the PageBox ID and log user messages
ResourcePrivilegedAction.java	resource loader
ServletLog.java	servlet to clear and display log
ServletStat.java	servlet to display stats
ServletUpdate.java	update servlet

Documentation:

- ❑ This document
- ❑ javadoc