Local Plugins: Third-party library isolation

Third-party libraries deployed and utilized by the vCenter Server appliance (VCSA) for its own needs that are currently exposed to partner plugins will be restricted and no longer available effectively from the next major release of vSphere.

In the versions of vSphere up to 7.0, the vSphere Client platform is not isolated. The local plugins have the possibility to import packages coming from third-party libraries deployed for the needs of vSphere Client platform. This is problematic for multiple reasons, such as:

- Changes to internal vSphere Client APIs could break plugin compatibility.
- Changes to a particular vSphere Client dependency (e.g. to consume security updates) could impact plugin compatibility.
- Historically plugins have been using a lot more 3rd party library dependencies than required (practically the only required ones were the Spring-related libs).

Plugins using vSphere Client internal dependencies present a serious risk from security and supportability point of view, since VMware must be able to:

- Update vSphere Client third-party libraries quickly if there is a notice about security vulnerability in any of them.
- Support each version of the Client for multiple years and therefore obliged to avoid unsupported library versions.

These are the essential reasons forcing VMware to make a step further in maintaining the reliability of the plugin integration model and make sure plugin developers are providing their own OSGi Java dependencies. Any logging implementation/facade, JSON/XML serialization, Apache utilities, etc. will have to be provided as part of the plugin: either as separate OSGi bundles or included in an existing plugin bundle's class path.

The library isolation of local plugins will take effect in two steps, starting with the release vSphere 7.0U3, when it will be turned off by default.

The library isolation will be turned on from the next major release of vSphere.

Overview

The isolation was achieved with the use of the Apache Aries library, that implements the OSGi Subsystem specification. A plugin represented as a Subsystem is installed as a Composite.
See: https://liferay.dev/blogs/-/blogs/osgi-subsystems-and-why-you-want-them

The exact OSGi capabilities shared from the vSphere Client Kernel region (aka the Platform) can be controlled with the use of Import-Package, Require-Bundle and Require-Capability statements in a generated SUBSYSTEM.MF file

The shared packages are: Public API packages, their dependencies, packages exported by the system bundle (packages present in the JVM, but not starting with java.*, also packages belonging to Java Servlet Spec)

Spring bootstrap is achieved by using reflection, so no linkage of Spring classes will be made, as the Spring classes will be different in the Platform and each plugin.

In this how-to document we will explore a step-by-step guide of how to comply with the new OSGi requirements.

Step-by-step guide on how to move to full isolated plugin (local setup)

Prerequisites: zip edit tool, telnet client, registered plugin, knowledge on the OSGi Console: see Using the OSGi console section here: https://www.vogella.com/tutorials/OSGi/article.html#access-to-the-osgi-console-for-your-eclipse-application

1. Place **osgi.fullIsolation.debug=true** in **webclient.properties**
2. Open with zip edit tool <address of your sdk>/server/webapps/h5-bridgewebapp.war
3. Remove the XML comments around console in /WEB-INF/web.xml
4. (Re)start server
5. Open your telnet client and connect to localhost, port 2401
6. The OSGi console opens up.
    a. The **diag** command is what we're after
    b. Use **ss** or **lb** to list currently deployed bundles
    c. Search for the bundle symbolic name of the plugins bundle (if using **ss** command)
7. Use **diag** <bundle id> to display the missing dependencies of your plugin's bundles.
8. If you're not certain which bundle/jar provides a package missing for your bundles you can use the **packages** <package name> command.
9. Add missing bundle dependencies in your **plugins** folder.
10. (Re)start server and repeat the Console procedure until all of your bundles successfully resolved their dependencies

Step-by-step guide on how to move to full isolated plugin (VCSA setup)

Prerequisites:

telnet client (Windows: Putty, Mac: use brew [brew install telnet], Linux: use package manager of choice to instal telnet)

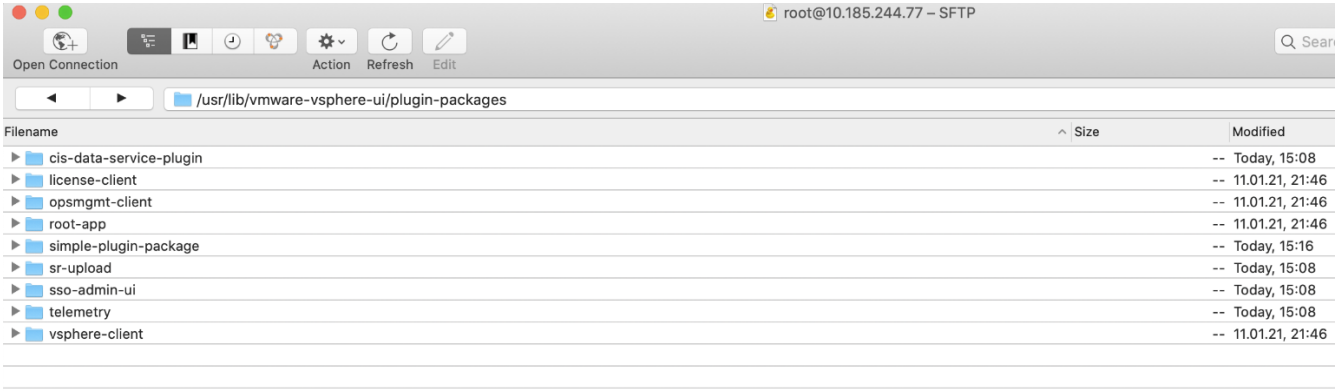
scp tool (Windows: WinSCP, Mac: Cyberduck or scp command, Linux: scp command)

basic vi knowledge http://heather.cs.ucdavis.edu/~matloff/UnixAndC/Editors/ViIntro.html

Warning: these steps should be undertaken only on a non-production test environment, that preferably doesn't have outside internet access. The steps could compromise security.

0. ssh to the test VCSA

1. Place your plugin files in **/usr/lib/vmware-vsphere-ui/plugin-packages** (we'll use the **simple-plugin-package** as example)



2. Make sure that the files of your plugin are owned by the vsphere-ui OS user

cd /usr/lib/vmware-vsphere-ui/plugin-packages/

chown vsphere-ui simple-plugin-package/ -R

The result should look like:

```
root@sc2-10-185-244-77 [ /usr/lib/vmware-vsphere-ui/plugin-packages ]# ls -lah
total 44K
drwxrwx--- 11 vsphere-ui root 4.0K Jan 15 13:16 .
drwxrwx---  7 vsphere-ui cis  4.0K Jan 15 13:27 ..
drwxrwx---  3 vsphere-ui root 4.0K Jan 15 13:26 cis-data-service-plugin
drwxrwx---  3 vsphere-ui root 4.0K Jan 11 19:46 license-client
drwxrwx---  3 vsphere-ui root 4.0K Jan 11 19:46 opsmgmt-client
drwxrwx---  3 vsphere-ui root 4.0K Jan 11 19:46 root-app
drwxr-xr-x  3 vsphere-ui root 4.0K Jan 15 13:26 simple-plugin-package
drwxrwx---  3 vsphere-ui root 4.0K Jan 15 13:26 sr-upload
drwxrwx---  3 vsphere-ui root 4.0K Jan 15 13:26 sso-admin-ui
drwxrwx---  3 vsphere-ui root 4.0K Jan 15 13:26 telemetry
drwxrwx---  3 vsphere-ui root 4.0K Jan 11 19:46 vsphere-client
```

3. Add **vsphere-ui-dependencies-available="false"** as an attribute in your plugin package manifest, aka plugin-package.xml and save the file.

```
<?xml version="1.0" encoding="UTF-8"?>

<pluginPackage id="com.vmware.o6jia.simple.fully.isolated"
      type="html"
      version="${core.plugins.version}"
      name="Sample plugin package fully isolated"
      description="Used for testing purposes"
      vendor="VMware"
      vsphere-ui-dependencies-available="false">

   <metadata>
      <property name="supportedServerVersions" value="6.7"/>
   </metadata>

   <bundlesOrder>
      <bundle id="com.vmware.o6jia.simple-lib" />
      <bundle id="com.vmware.o6jia.spring-xml-consumer-isolated" />
      <bundle id="com.vmware.o6jia.simple-front-end-isolated" />
   </bundlesOrder>

</pluginPackage>
```

4. Add **osgi.fullIsolation.debug=true** in webclient.properties

vi /etc/vmware/vsphere-ui/webclient.properties

```
live.updates.navtree.enabled = true
live.updates.lists.enabled = true
live.updates.objectstate.enabled = true

osgi.fullIsolation.debug=true

# Web client session timeout in minutes, default is 120, i.e. 2 hours
```

5. Enable the OSGi Console

5.1: **cd /usr/lib/vmware-vsphere-ui/server/webapps**

5.2: **unzip -j "h5-bridge-webapp.war" "WEB-INF/web.xml" -d "/usr/lib/vmware-vsphere-ui/server/webapps/WEB-INF"** (this will unzip only the /WEB-INF/web.xml file)

Edit the WEB-INF/web.xml by removing the the XML comments around the init-param "commandline"

5.3: **vi WEB-INF/web.xml**

The web.xml should look like this:

```
<servlet id="bridge">
   <servlet-name>equinoxbridgeservlet</servlet-name>
   <servlet-class>com.vmware.vsphere.bridge.BridgeServletEx</servlet-class>
   <!-- Telnet to port 2401 for osgi console -->


   <init-param>
      <param-name>commandline</param-name>
      <param-value>-console 2401</param-value>
   </init-param>
```

Save the file. Then modify the h5-bridge-webapp.war with the modified web.xml

5.4: **zip -u h5-bridge-webapp.war WEB-INF/web.xml**

```
root@sc2-10-185-244-77 [ /usr/lib/vmware-vsphere-ui/server/webapps ]# zip -u h5-bridge-webapp.war WEB-INF/web.xml
updating: WEB-INF/web.xml (deflated 76%)
```

Be sure that the output of the zip command is "updating", not adding. Check filesystem permissions, using **ls -lah**, likely the h5-bridge-webapp.war is now not owned by vsphere-ui.

5.5: **chown vsphere-ui h5-bridge-webapp.war**

6. Open ports on the VCSA:

iptables -A INPUT -j ACCEPT

7. Restart the vsphere-ui:

service-control --restart vsphere-ui

8. Telnet to <VCSA IP/hostname> port 2401

telnet <vc_ip> 2401

```
proynovd@proynovd-a03 ~ % telnet 10.185.244.77 2401
Trying 10.185.244.77...
Connected to sc2-10-185-244-77.eng.vmware.com.
Escape character is '^]'.

____________________________
Welcome to Apache Felix Gogo

g!
```

9. Use the **ss** command to list all bundles and look for the bundles belonging to your plugin

```
272     ACTIVE      org.osgi.service.subsystem.region.context.1_1.0.0
273     RESOLVED    com.vmware.o6jia.simple-lib_6.1.0.SNAPSHOT
274     INSTALLED   com.vmware.o6jia.spring-xml-consumer_7.0.2.00000
275     INSTALLED   org.eclipse.virgo.web.dm_1.0.0
276     INSTALLED   com.vmware.o6jia.simple-front-end_7.0.2.00000
```

We can observe that the bundles of your plugin are in the INSTALLED state, meaning they weren't resolved and could not be started.

10. Use **diag <bundle id>** to diagnose missing dependencies

```
g! diag 274
com.vmware.o6jia.spring-xml-consumer [274]
  Unresolved requirement: Import-Package: org.aopalliance.aop
  Unresolved requirement: Import-Package: com.vmware.o6jia.context
    -> Export-Package: com.vmware.o6jia.context; bundle-symbolic-name="org.eclipse.virgo.web.dm"; bundle-version="1.0.0"; version="0.0.0"
       org.eclipse.virgo.web.dm [275]
         Unresolved requirement: Import-Package: org.springframework.aop
```

The command will list all unresolved Import-Package/Require-Bundle requirements.

11. Add Spring Framework

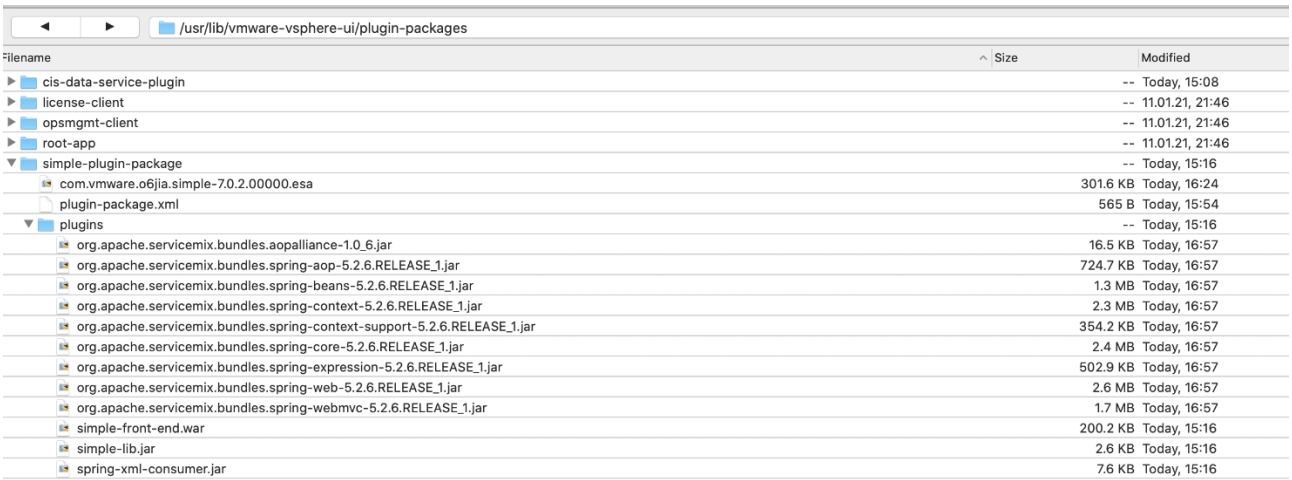
Considering that the sample plugin uses a Spring MVC controllers so we can take a shortcut and add the following dependencies in our plugins folder

Spring MVC

If you use Spring MVC controllers in your plugin the following list provides a somewhat latest picture of the bundles required to resolve Spring MVC dependencies:

- org.apache.servicemix.bundles.aopalliance-1.0_6
- org.apache.servicemix.bundles.spring-aop-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-beans-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-context-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-context-support-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-core-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-expression-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-web-5.2.6.RELEASE_1
- org.apache.servicemix.bundles.spring-webmvc-5.2.6.RELEASE_1

The plugin's folder will look like:



12. Restart vsphere-ui, so we can make another iteration and see any other missing dependencies

service-control --restart vsphere-ui

13. Telnet will disconnect. Connect again to the OSGi console:

telnet <vc_ip> 2401

This time our plugin looks like the following:

```
272     ACTIVE      org.osgi.service.subsystem.region.context.1_1.0.0
273     RESOLVED    org.apache.servicemix.bundles.spring-core_5.2.6.RELEASE_1
274     RESOLVED    org.apache.servicemix.bundles.spring-context-support_5.2.6.RELEASE_1
275     RESOLVED    com.vmware.o6jia.simple-lib_6.1.0.SNAPSHOT
276     RESOLVED    com.vmware.o6jia.spring-xml-consumer_7.0.2.00000
277     INSTALLED   com.vmware.o6jia.simple-front-end_7.0.2.00000
278     RESOLVED    org.eclipse.virgo.web.dm_1.0.0
279     RESOLVED    org.apache.servicemix.bundles.aopalliance_1.0.0.6
280     RESOLVED    org.apache.servicemix.bundles.spring-web_5.2.6.RELEASE_1
281     RESOLVED    org.apache.servicemix.bundles.spring-beans_5.2.6.RELEASE_1
282     RESOLVED    org.apache.servicemix.bundles.spring-context_5.2.6.RELEASE_1
283     RESOLVED    org.apache.servicemix.bundles.spring-expression_5.2.6.RELEASE_1
284     RESOLVED    org.apache.servicemix.bundles.spring-aop_5.2.6.RELEASE_1
285     RESOLVED    org.apache.servicemix.bundles.spring-webmvc_5.2.6.RELEASE_1
```

We can observe that only one war is not resolved. We'll repeat steps 10

14. Use **diag <bundle id>** to diagnose remaining missing dependencies

```
g! diag 276
com.vmware.o6jia.simple-front-end [276]
  Unresolved requirement: Import-Package: com.fasterxml.jackson.core
```

Note: that the id of the simple-front-end bundle is now different after the last restart of the vsphere-ui. That is due to the random nature in which plugin bundles are installed in the OSGi runtime.

15. If you're not sure which bundle/jar provides a package that your plugin requires you can use the following OSGi console command:

packages <required package>

Typically the bundle-symbolic-name is going to be a good orientation point to bundle that provides a required package. In this case if we make a search for "jackson-core", we'll likely hit the jar that is required. https://search.maven.org/search?q=jackson-core

16. Continue adding required bundles in your plugins dir until all your plugin bundles are resolved.

17. Make the necessary changes to your build.

Note 1: Some jars, posted in the Maven Central are not OSGi-fied, or they are not OSGi bundles. If you happen to require such a jar, then you'll have to OSGi-fy it as part of your build. You can use Spring's bundlor or BND:

https://docs.spring.io/s2-bundlor/1.0.x/user-guide/htmlsingle/user-guide.html

https://github.com/bndtools/bnd/tree/master/maven/bnd-maven-plugin

Some jars have different artifact id to denote that it's an OSGi bundle like Apache's http client:

https://search.maven.org/artifact/org.apache.httpcomponents/httpclient-osgi

vs

https://search.maven.org/artifact/org.apache.httpcomponents/httpclient

Note 2: If your plugin requires the following packages:

com.vmware.vise.messaging.endpoints
com.vmware.vise.messaging.remoting

you should remove them. They are not needed.

They are also not exported in the full isolation mode.

Note 3: OSGi-fied Spring and other libs are provided by Apache ServiceMix,
see: https://mvnrepository.com/artifact/org.apache.servicemix.bundles

VMware recommends migrating directly to a Remote plugin which does not require the afore mentioned changes.