Java Properties in CAPS 5.1
Using Properties for Runtime Control, Note 1

Access Properties in a Java Collaboration Definition

Introduction

Java CAPS 5.1 developer may need to determine a runtime value of one or more system properties to, perhaps, use its vale for runtime flow control. This is the first in a series of notes on the use of Java Properties for controlling runtime behaviour of Java CAPS solutions.

This paper walks through an example of configuring Java CAPS 5.1 and developing, and exercising, a Java Collaboration Definition that dumps all system properties to a file and that accesses the values of two specific properties, by name, and dumping them to a file as well.

The collaboration will be triggered by a JMS trigger message whose content it will ignore.

Configure server.policy

The following assumes the use of Sun SeeBeyond Integration Server, bundled with Java CAPS 5.1.3. If you use a different application server you may not need to change security configuration or you may need to do it differently.

By default Java CAPS runtime domain is configured to allow read access to properties. Empirical evidence suggests that it is not enough. One, it seems, must have read and write access to get system properties.

Modify
<JCAPSRoot>/is/domain/<yourdomaim>/config/server.policy
replacing

```
permission java.util.PropertyPermission "*", "read";
```

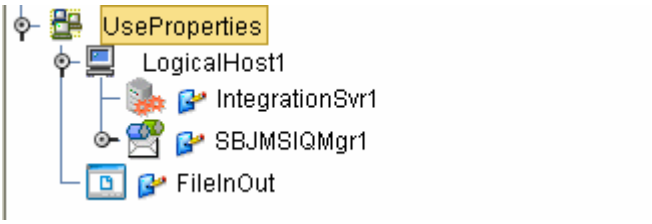
with

```
permission java.util.PropertyPermission "*", "read,write";
```

If your domain is running it will need to be re-started to make this change effective.

Java CAPS Environment

The Java CAPS Environment, required to deploy and execute the example, will consist of Sun SeeBeyond Integration Server, a Sun SeeBeyond JAM IQ Manager and a File External System.



There are no non-standard configuration settings.

Create Java CAPS Project

The project, GetProperties, will contain a single Java Collaboration Definition, jcdGetProperties, and associated connectivity map, its objects and a deployment profile.

The JCD will be triggered by arrival of a message to a JMS Queue and will write output to a file using the File eWay.

Create a Java Collaboration Definition, choose a JMS Receive operation, add FileClient message and add appropriate Java statements.

```
public void
    receive
        ( com.stc.connectors.jms.Message input
        , com.stc.connector.appconn.file.FileApplication W_toFileClient )
    throws Throwable
{
    // need to configure server.policy for the domain
    // permission java.util.PropertyPermission "*", "read,write";
    ;
    java.util.Properties p = System.getProperties();
    ;
    // get system properties list as string
```

```
    java.io.ByteArrayOutputStream baos
                          = new java.io.ByteArrayOutputStream();
    java.io.PrintStream ps = new java.io.PrintStream( baos );
    p.list( ps );
    ps.flush();
    baos.flush();
    String sSystemProperties = baos.toString();
    ;
    // write system properties string
    W_toFileClient.setText( sSystemProperties );
    W_toFileClient.write();
    ;
    // write classpath property
    String javaclasspath = p.getProperty( "java.class.path" );
    W_toFileClient.setText( "\njava.class.path\n" + javaclasspath );
    W_toFileClient.write();
    ;
    // write boot classpath property
    String bootclasspath = p.getProperty( "sun.boot.class.path" );
    W_toFileClient.setText( "\nsun.boot.class.path\n" + bootclasspath );
    W_toFileClient.write();
    ;
}
```

System.getProperties returns a Properties object containing all system properties.

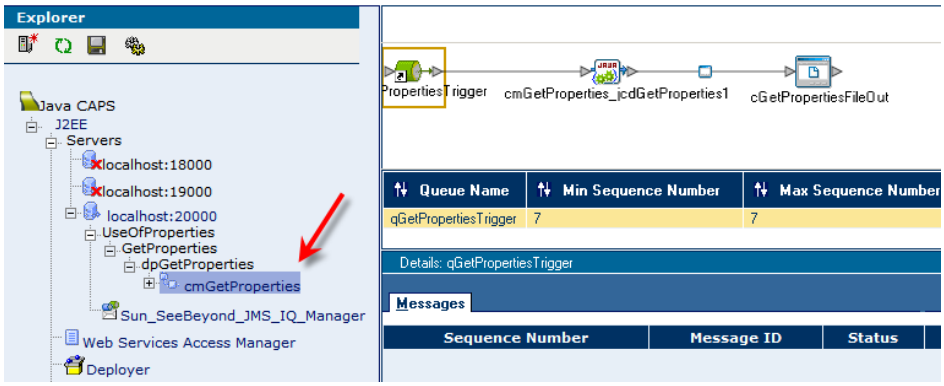
Manipulation of ByteArrayOutputStream and PrintStream allows us to use the list method of the Properties object to get a string equivalent of the properties list to write to the file.

getProperty method of the Properties object returns the vale of a single named property as a String.

All strings are appended to the same file because, by default, the File connector in the Connectivity Map, has the property “Multiple Records per file” set to true. Setting this property to false would cause each write to create a new file.

Once the JCD and CM, see above, are created, it is time to create a deployment profile, build and deploy the project.

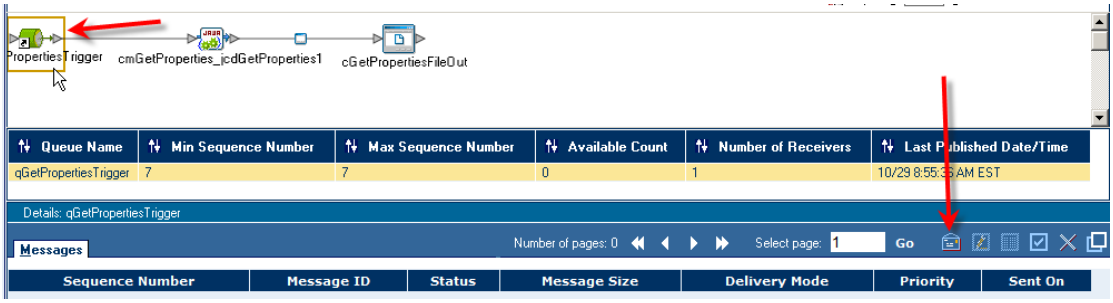
Deployed project will show up in the Enterprise Manager.



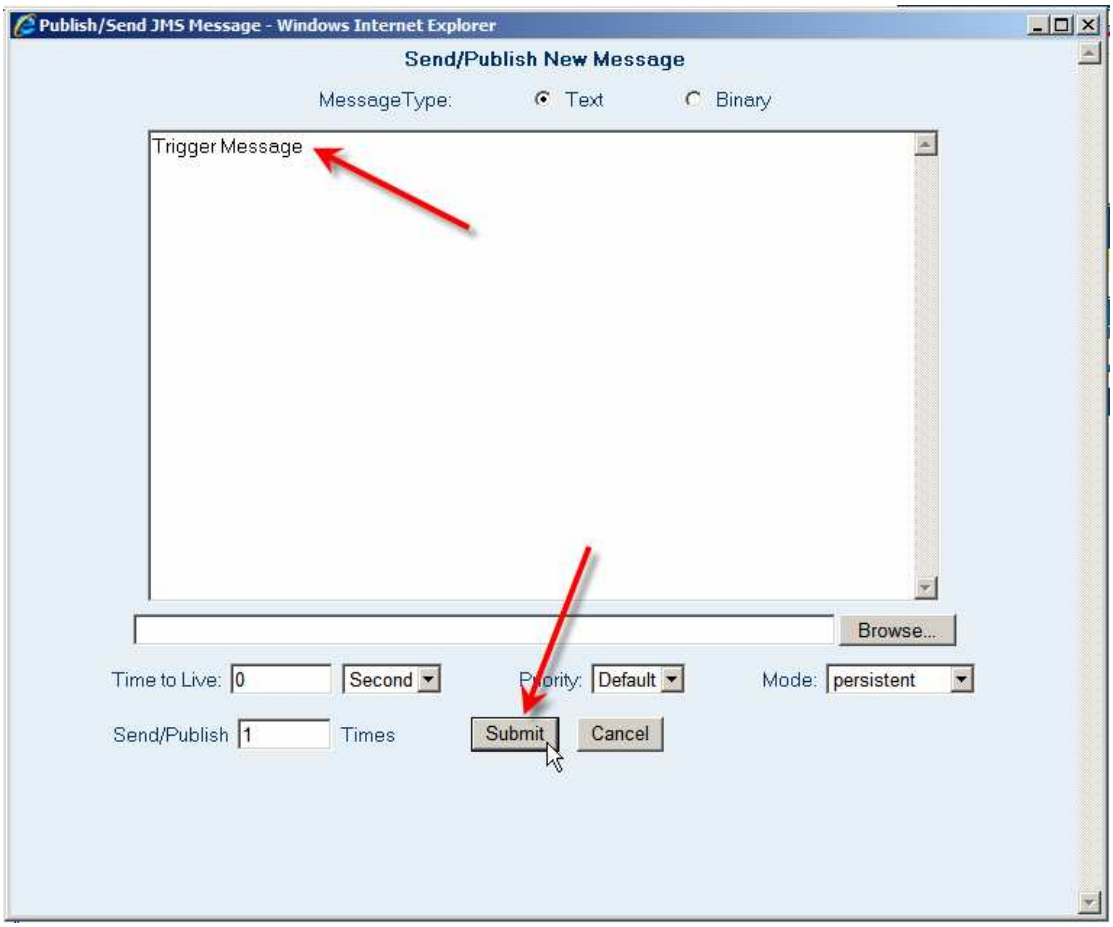
Exercise Project

To exercise the project we need to manually publish a JMS Message to the JMS Queue qGetPropertiesTrigger.

Click the Queue object in the connectivity map graphic in the Enterprise Manager then click the "Send/Publish" toolbar button.



Enter some string in the dialogue window and click the Submit button.



If all is well the collaboration will execute and will produce a file in the output directory. The file will contain something similar to the following, with some content omitted for brevity.

```
-- listing properties --
java.vendor=Sun Microsystems Inc.
com.sun.aas.imqBin=C:\jcaps513\logicalhost\is\imq\bin
javax.xml.soap.SOAPFactory=com.sun.xml.messaging.saaj.soap.ver1_...
org.xml.sax.parser=org.xml.sax.helpers.XMLReaderAdapter
catalina.base=C:\JCAPS513\logicalhost\is\domains\props
sun.management.compiler=HotSpot Server Compiler
server.name=server
.
.
.
file.encoding=Cp1252
org.omg.CORBA.ORBClass=com.sun.corba.ee.impl.orb.ORBImpl
com.sun.CORBA.ORBUserConfigurators.com.sun.enterprise.iiop.PEORBConfigurator=
java.specification.version=1.5
javax.net.ssl.trustStore=C:/JCAPS513/logicalhost/is/domains/pr...
javax.xml.transform.TransformerFactory=com.sun.org.apache.xalan.internal.xsl...
javax.net.ssl.keyStorePassword=changeit


java.class.path
C:\JCAPS513\logicalhost\is\lib\commons-
launcher.jar;C:\jcaps513\logicalhost\is\lib\activation.jar;C:\jcaps513\logicalhost\is\
lib\admin-cli.jar;C:\jcaps513\logicalhost\is\lib\appserv-
admin.jar;C:\jcaps513\logicalhost\is\lib\appserv-
cmp.jar;C:\jcaps513\logicalhost\is\lib\appserv-
ext.jar;C:\jcaps513\logicalhost\is\lib\appserv-
jstl.jar;C:\jcaps513\logicalhost\is\lib\appserv-
rt.jar;C:\jcaps513\logicalhost\is\lib\com.stc.icu4j.jar;C:\jcaps513\logicalhost\is\lib
\com.stc.jms.stcjms.jar;C:\jcaps513\logicalhost\is\lib\com.stc.jms.stcqueueviewer.jar;
C:\jcaps513\logicalhost\is\lib\com.stc.jmsjca.core.jar;C:\jcaps513\logicalhost\is\lib\
com.stc.jmsmx.core.jar;C:\jcaps513\logicalhost\is\lib\com.stc.jmsmx.sjsmq.jar;C:\jcaps
513\logicalhost\is\lib\com.stc.jmsmx.stcms.jar;C:\jcaps513\logicalhost\is\lib\com.stc.
jmsmx.wave.jar;C:\jcaps513\logicalhost\is\lib\com.stc.wave.lwmsClient.jar;C:\jcaps513\
logicalhost\is\lib\com.stc.wave.mgmt.jar;C:\jcaps513\logicalhost\is\lib\com.stc.wave.w
ave.jar;C:\jcaps513\logicalhost\is\lib\commons-
logging.jar;C:\jcaps513\logicalhost\is\lib\j2ee-
svc.jar;C:\jcaps513\logicalhost\is\lib\j2ee.jar;C:\jcaps513\logicalhost\is\lib\jax-
qname.jar;C:\jcaps513\logicalhost\is\lib\jaxr-
api.jar;C:\jcaps513\logicalhost\is\lib\jaxr-
impl.jar;C:\jcaps513\logicalhost\is\lib\jaxrpc-
api.jar;C:\jcaps513\logicalhost\is\lib\jaxrpc-
impl.jar;C:\jcaps513\logicalhost\is\lib\mail.jar;C:\jcaps513\logicalhost\is\lib\net.sf
.hulp.meas.impl.jar;C:\jcaps513\logicalhost\is\lib\relaxngDatatype.jar;C:\jcaps513\log
icalhost\is\lib\saaj-api.jar;C:\jcaps513\logicalhost\is\lib\saaj-
impl.jar;C:\jcaps513\logicalhost\is\lib\stcjms_453.jar;C:\jcaps513\logicalhost\is\lib\
xercesImpl.jar;C:\jcaps513\logicalhost\is\lib\xsdlib.jar;C:/JCAPS513/logicalhost/jre/l
ib/tools.jar;C:/jcaps513/logicalhost/is/lib/install/applications/jmsra/imqjmsra.jar;C:
/jcaps513/logicalhost/is/imq/lib/jaxm-
api.jar;C:/jcaps513/logicalhost/is/imq/lib/fscontext.jar;C:/jcaps513/logicalhost/is/li
b/ant/lib/ant.jar;C:/jcaps513/logicalhost/is/lib/ant/lib/ant-
launcher.jar;C:/jcaps513/logicalhost/is/derby/derby.jar

sun.boot.class.path
C:/jcaps513/logicalhost/is/lib/endorsed\dom.jar;C:/jcaps513/logicalhost/is/lib/endorse
d\xalan.jar;C:/jcaps513/logicalhost/is/lib/endorsed\xercesImpl.jar;C:\JCAPS513\logical
host\jre\lib\rt.jar;C:\JCAPS513\logicalhost\jre\lib\i18n.jar;C:\JCAPS513\logicalhost\j
re\lib\sunrsasign.jar;C:\JCAPS513\logicalhost\jre\lib\jsse.jar;C:\JCAPS513\logicalhost
\jre\lib\jce.jar;C:\JCAPS513\logicalhost\jre\lib\charsets.jar;C:\JCAPS513\logicalhost\
jre\classes
```

Note the list of property name/value pairs, under --listing properties-- heading.

Note, too, java.class.path and sun.boot.class.path listed separately. The three entries correspond to the three write statements in the JCD.

Closing

The JCD developed in this note shows how a complete set of Java system properties as well as individual system properties can be obtained by a Java CAPS Java Collaboration Definition at runtime.