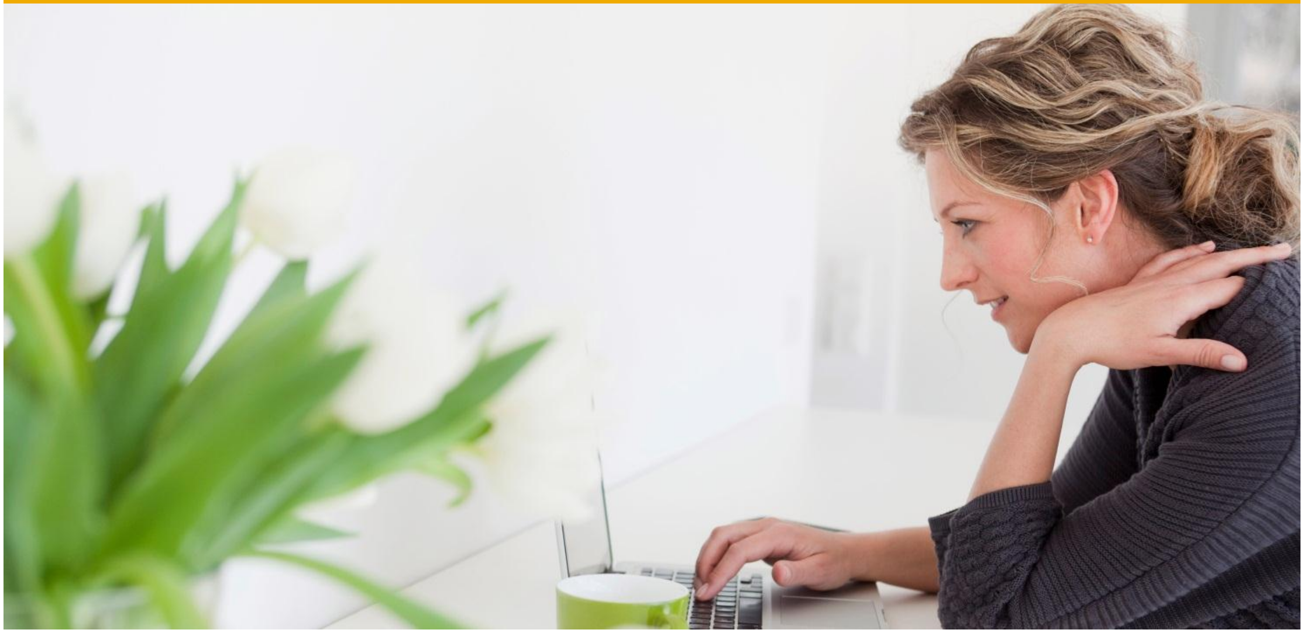


LAB4b: Channel Manager

Implement a New Channel

TABLE OF CONTENTS

AGENDA

1. Implement the Channel
2. Configure the Channel
3. Test the Channel

BEFORE YOU START…

Please start the Mobiliser 5.1 Lab Virtual machine.

The Login with user is "mobiliser" and password "sybase".

IMPLEMENT THE CHANNEL

This lab will implement an asynchronous channel to write outbound messages to files. These files can be read, parsed and made available to the Mobiliser.

1. Open package com.sybase365.mobiliser.custom.project.channels and add a FileChannel class.

```
package com.sybase365.mobiliser.custom.project.channels;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.LinkedList;
import java.util.List;
import java.util.Map.Entry;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

import com.sybase365.mobiliser.util.messaging.api.Message;
import com.sybase365.mobiliser.util.messaging.api.email.EmailMessage;
import com.sybase365.mobiliser.util.messaging.api.sms.SmsMessage;
import com.sybase365.mobiliser.util.messaging.api.ussd.UssdInputMessage;
import com.sybase365.mobiliser.util.messaging.api.ussd.UssdSelectMessage;
import com.sybase365.mobiliser.util.messaging.api.ussd.UssdTextMessage;
import
com.sybase365.mobiliser.util.messaging.channelmanager.api.AsynchronousSendReceiveChannel;
import
com.sybase365.mobiliser.util.messaging.channelmanager.api.callbacks.AsynchronousChannelReceiv
eCallback;
import com.sybase365.mobiliser.util.messaging.channelmanager.api.exceptions.ChannelException;
import com.sybase365.mobiliser.util.messaging.template.api.IMessagingEngine;
```

```
public class FileChannel implements AsynchronousSendReceiveChannel,
              InitializingBean, DisposableBean {

       static Logger LOG = LoggerFactory.getLogger(FileChannel.class);

       private String channelId;

       private String destinationId;

       private String channelFolder;

       private AsynchronousChannelReceiveCallback callback;

       private IMessagingEngine messagingEngine;

       private FileChannelPoller poller;
```

```
@Override
public void send(Message message) throws ChannelException {
       File f = new File(getChannelFolder() + File.separator
                     + message.getRecipient().getAddress());
       BufferedWriter writer = null;
       try {
       LOG.trace("Received message {}", message);
       String messageText = "###";
       if (message instanceof SmsMessage) {
              messageText = new String(((SmsMessage) message).getText()
              .getContent(), ((SmsMessage) message).getText()
              .getCharset());
       } else if (message instanceof UssdInputMessage) {
              messageText = ((UssdInputMessage) message).getLabel();
       } else if (message instanceof UssdSelectMessage) {
              messageText = ((UssdSelectMessage) message).getTitle() + ":";
              for (Entry<Integer, String> e : ((UssdSelectMessage) message)
                            .getOptions().entrySet()) {
                     messageText += "'" +e.getKey()+":"+ e.getValue() + "' ";
              }
       } else if (message instanceof UssdTextMessage) {
              messageText = new String(((UssdTextMessage) message).getText()
              .getContent(), ((UssdTextMessage) message).getText()
              .getCharset());
       } else if (message instanceof EmailMessage) {
              messageText = new String(((EmailMessage) message).getText()
              .getContent(), ((EmailMessage) message).getText()
              .getCharset());
       }
       writer = new BufferedWriter(new OutputStreamWriter(
              new FileOutputStream(f, true)));
       writer.write("in:" + message.getSender().getAddress() + ":"
                     + messageText + "\n");
       LOG.debug("Message written to file {}", f.getAbsolutePath());
       } catch (FileNotFoundException e) {
              LOG.error("Couldn't write message to file", e);
       } catch (IOException e) {
              LOG.error("Couldn't write message to file", e);
```

```
		} finally {
			try {
				if (writer != null) {
					writer.close();
				}
			} catch (IOException e) {
				LOG.error("Couldn't write message to file", e);
			}
		}
	}
	@Override
	public void afterPropertiesSet() throws Exception {
		this.poller = new FileChannelPoller();
		this.poller.start();

		LOG.info("File poller started");
	}

	@Override
	public void destroy() throws Exception {
		this.poller.shutdown();

		LOG.info("File poller closed down");
	}

	@Override
	public boolean supports(Message message) {
		return true;
	}

	@Override
	public void setReceiveCallback(AsynchronousChannelReceiveCallback callback) {
		this.callback = callback;
	}

	public void setChannelId(String channelId) {
		this.channelId = channelId;
	}

	@Override
	public String getChannelId() {
		return this.channelId;
	}
	public void setDestinationId(String destinationId) {
		this.destinationId = destinationId;
	}
	public void setMessagingEngine(IMessagingEngine engine) {
		this.messagingEngine = engine;
	}

	public String getChannelFolder() {
		return this.channelFolder;
	}

	public void setChannelFolder(String channelFolder) {
		this.channelFolder = channelFolder;
	}
```

```java
private class FileChannelPoller extends Thread {
	private boolean run = true;

	public FileChannelPoller() {
		super();
	}

	@Override
	public void run() {
		while (this.run) {
			File channelDir = new File(FileChannel.this.getChannelFolder());
LOG.trace("File poller scans directory '{}' for new messages now",
		channelDir.getAbsolutePath());

	if (channelDir.listFiles() != null) {
		for (File f : channelDir.listFiles()) {
			LOG.trace("Inspecting file '{}'", f.getAbsolutePath());
			final String sender = f.getName();
			final List<String> inMessages = new LinkedList<String>();
			BufferedReader reader = null;
			BufferedWriter writer = null;
			try {
				reader = new BufferedReader(new FileReader(f));
				writer = new BufferedWriter(new OutputStreamWriter(
						new FileOutputStream(f, true)));
				String line = reader.readLine();
				while (line != null) {
					if (line.equals("PROCESSED")) {
						inMessages.clear();
					}
					if (line.startsWith("out:")) {
						inMessages.add(line.substring(4));
					}
					line = reader.readLine();
				}
				if (inMessages.size() > 0) {
					writer.write("PROCESSED\n");
				}
			} catch (FileNotFoundException e) {
				LOG.error("Couldn't scan file", e);
			} catch (IOException e) {
				LOG.error("Couldn't scan file", e);
			} finally {
				try {
					if (reader != null) {
						reader.close();
					}
					if (writer != null) {
						writer.close();
					}
				} catch (IOException e) {
					LOG.error("Couldn't scan file", e);
				}
			}
```

```
        LOG.trace("Processing messages now");
        for (final String inMessage : inMessages) {
                if (inMessage.indexOf(":") == -1) {
                LOG.error("Recipient phone number missing for line '{}'",
                        inMessage);
                }

                Message message = messagingEngine
                                .parseSimpleTextMessage("sms",
                                                inMessage.substring(inMessage
                                                                .indexOf(":") + 1));

                message.setSender(sender);
                final String recipient = inMessage.substring(0,
                                inMessage.indexOf(":"));
                message.setRecipient(recipient);
                FileChannel.this.callback.receiveMessage(message,
                                channelId, destinationId);

                LOG.debug("Sent message to '{}' on queue '{}'",
                                        recipient, destinationId);
                                }
                        }
                }

                try {
                        Thread.sleep(500);
                } catch (InterruptedException e) {
                        return;
                }
                if (Thread.interrupted()) {
                        return;
                }
                }
        }

        public void shutdown() {
                this.run = false;
        }
}

}
```

2. In src/main/resources add a new subfolder com/sybase365/mobiliser/util/messaging/channelmanager/file and create these two files in it.

default-properties.properties

```
# folder where to store communication files
channelFolder=/home/mobiliser/messages
channelId=defaultQ
destinationId=inQ
```

spring-beans.xml

```
<?xml version="1.0" encoding="utf-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

  <bean id="fileChannel" class="com.sybase365.mobiliser.custom.project.channels.FileChannel">
    <property name="channelId" value="${channelId}" />
    <property name="destinationId" value="${destinationId}" />
    <property name="channelFolder" value="${channelFolder}" />
    <property name="messagingEngine" ref="messagingEngine" />
    <property name="receiveCallback" ref="asyncReceiveCallback" />
  </bean>
</beans>
```

(It is advisable to replace the double-quotes in Eclipse if you paste this into your Virtual Machine)

3. Add the channel factory configuration to the bundle-context.xml

```
(…)
<bean id="fileChannelFactory"
    class="com.sybase365.mobiliser.util.messaging.channelmanager.util.impl.ChannelFactoryBean
    ">
    <property name="beanName" value="fileChannel" />
    <property name="dependentBeans">
     <util:map>
       <entry key="messagingEngine" value-ref="messagingEngine" />
       <entry key="asyncReceiveCallback" value-ref="asyncReceiveCallback" />
     </util:map>
    </property>
    <property name="resources">
     <list>
      <value>classpath:/com/sybase365/mobiliser/util/messaging/channelmanager/file/spring-
    beans.xml</value>
     </list>
    </property>
    <property name="defaultProperties">
     <bean class="org.springframework.beans.factory.config.PropertiesFactoryBean">
       <property name="location"
    value="classpath:/com/sybase365/mobiliser/util/messaging/channelmanager/file/default-
    properties.properties" />
      </bean>
    </property>
</bean>

(…)
```

4. Expose the channel factory to the service registry in the bundle-context-osgi.xml

```
    <osgi:service ref="fileChannelFactory"

interface="com.sybase365.mobiliser.util.messaging.channelmanager.util.IChannelFactoryBean">
        <osgi:service-properties>
           <beans:entry key="channelType" value="file" />
        </osgi:service-properties>
    </osgi:service>

   <osgi:reference id="asyncReceiveCallback"

interface="com.sybase365.mobiliser.util.messaging.channelmanager.api.callbacks.AsynchronousCh
annelReceiveCallback" />
```

5. If necessary, edit the POM.xml file in the channelManager project by importing the package shown below. The Import-Package tag should look as follows:

```
        <Import-Package>
               com.sybase365.mobiliser.util.messaging.channelmanager.util
             ,*
       </Import-Package>
```

BUILD THE NEW CHANNEL

1. Because the "com.sybase365.mobiliser.custom.project.channel" bundle is the only one being modified, we can use a "hot deployment" procedure. This will allow updated bundles to be moved to the runtime environment without bringing down the Mobiliser money server.
2. Open a terminal (Applications/Accessories/Terminal)
3. Change to the following directory
   ```
   $ cd ~/workspace/custom/channels
   ```
4. Run following command to build the new channel bundle.
   ```
   $ mvn clean install
   ```
5. Once the maven shows "**BUILD SUCCESS**", go to the following directory:
   ```
   $ cd ~/workspace/custom/channels/target
   ```
6. You should see following two files among others:
   ```
   $ ls -l
   com.sybase365.mobiliser.custom.project.channels-1.2.0-SNAPSHOT.jar
   com.sybase365.mobiliser.custom.project.channels-1.2.0-SNAPSHOT-sources.jar
   ```

We will be deploying the "*com.sybase365.mobiliser.custom.project.channels-1.2.0-SNAPSHOT.jar"* bundle.

INSTALL CHANNEL BUNDLE – HOT DEPLOYMENT

Bundle Build directory:
~/workspace/custom/channels/target
Runtime Bundle Source directory:
~/workspace/custom/dist/target/com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT/money/bundles/14-mobiliser-messaging

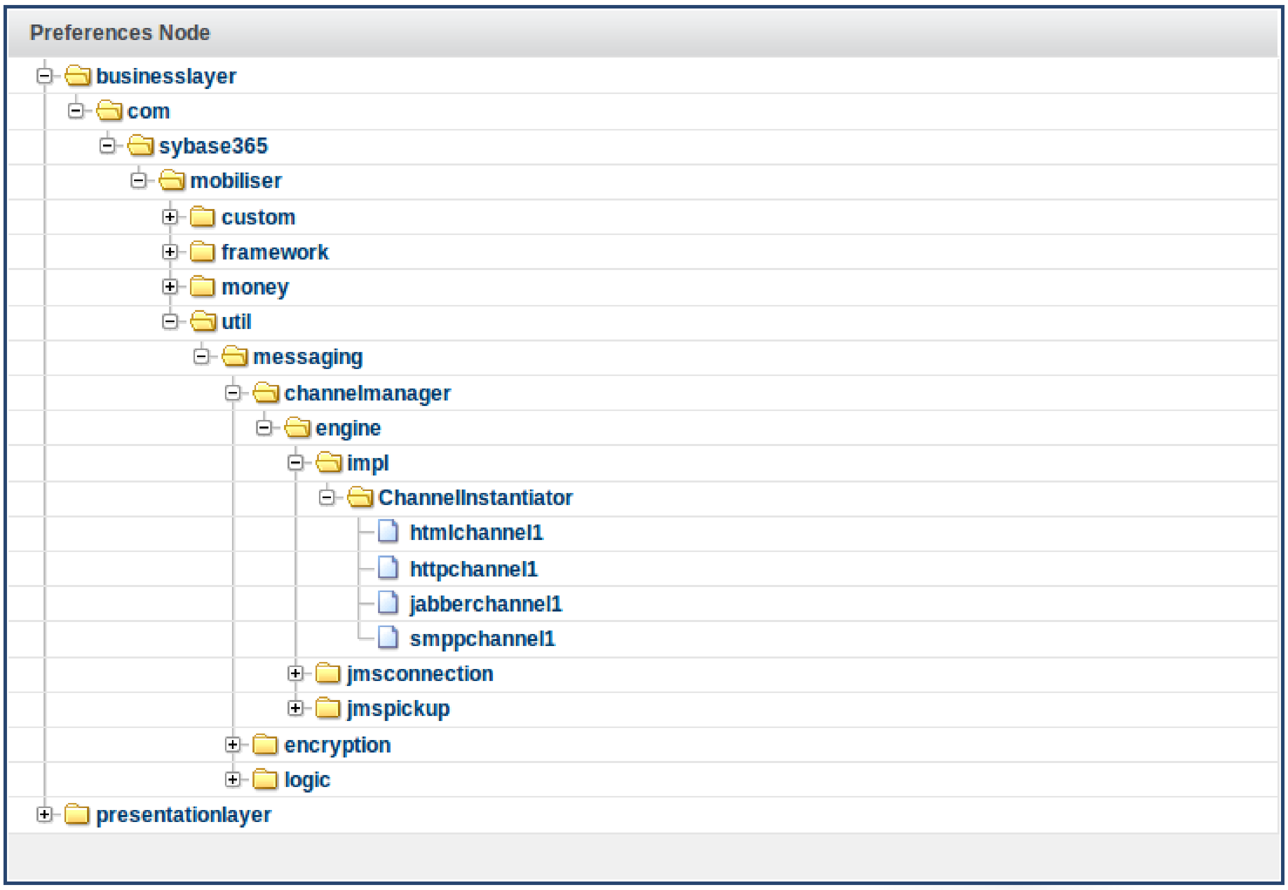
Use the same technique as was demonstrated for the payment handler deployment in Lab3a (See INSTALL PAYMENT HANDLER BUNDLE – HOT DEPLOYMENT) to deploy channel manager – complete the remaining steps before testing it.

In the AIMS Console look for bundle ***com.sybase365.mobiliser.custom.projects.channels (Filter for Project Channels)***, Stop, Update and Activate the bundle. There should be two service Id's that are IChannelFactoryBeans, verify that one is the new FileChannel.

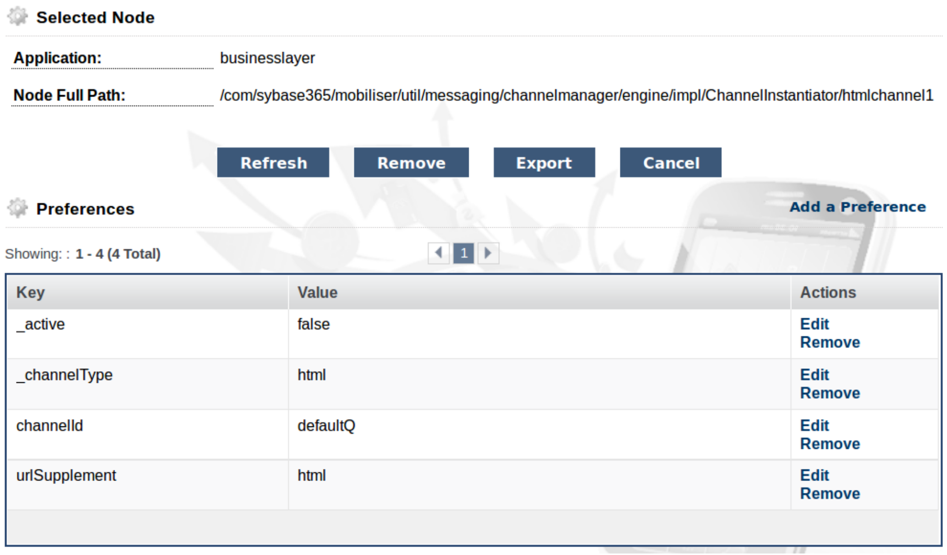
Configure the Channel
We will next change the default channel implementation from the HTML channel to the new File Channel

1. Log on to the Mobiliser dashboard at http://localhost:8082/portal using the opsmgr user (secret)

2. Open the preferences screen and navigate to ChannelInstantiator



3. Deactivate the HTML channel – by resetting "_active" to "false" (current value is true). Click on edit to do this.



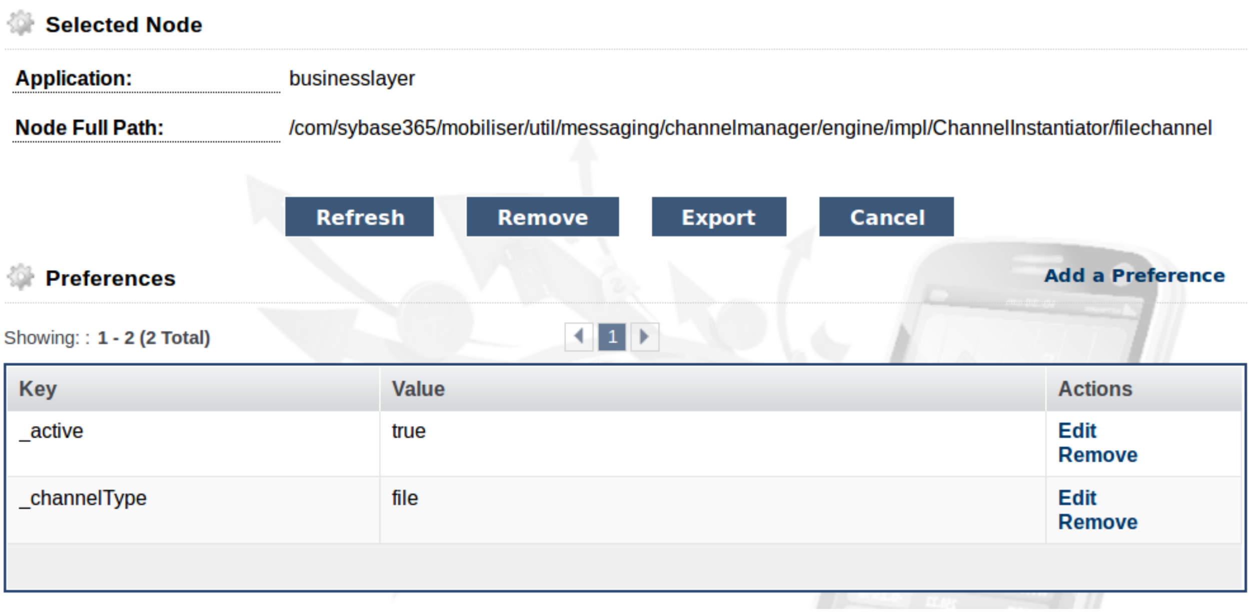
Application:	businesslayer
Node Full Path:	/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/ChannelInstantiator/htmlchannel1

Showing: : **1 - 4 (4 Total)**

Key	Value	Actions
_active	false	Edit Remove
_channelType	html	Edit Remove
channelId	defaultQ	Edit Remove
urlSupplement	html	Edit Remove

4. Create the “filechannel” using “Add a Preference Node” from the Menu option.

5. Add these new preferences to activate the File Channel (New Preferences)
The node path**:**
/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/ChannelInstantiator/filechannel


Selected Node

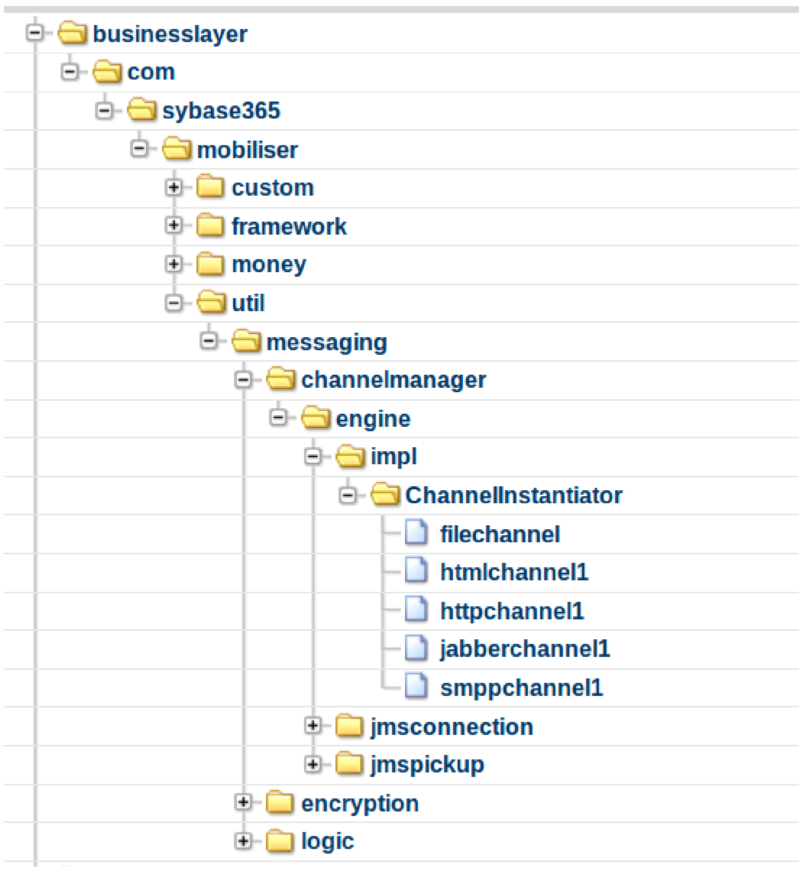
Application:	businesslayer
Node Full Path:	/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/ChannelInstantiator/filechannel

Preferences

Showing: : **1 - 2 (2 Total)**

Key	Value	Actions
_active	true	Edit Remove
_channelType	file	Edit Remove

6. After page refresh you should see this:



Test the Channel

1. Log on to the Mobiliser CST at http://localhost:8082/portal using the cstfull user
2. Navigate to the “NOTIFICATION MANAGER” → “Find Edit/Message” option
3. On the “Find Message” page click on the “Search” button.
4. Find the row with the name “CUSTOMER_MSISDN_OTP”. Click the “Test” link on the same row under the “Actions” column.
5. Enter a test MSISDN in the “Receiver” textbox. Add a text message in the “otp” textbox. Click the “Run Test” button to send the message.
6. Look in the "/home/mobiliser/tmp" directory and verify a file has been created. The name of the file will be the value entered as the "Receiver" on the “Test Message” page. Its contents should be the string enter as the "opt" value

T**roubleshooting:**

Create the folder/directory that you referenced in default-properties.properties, e.g. $mkdir /home/mobiliser/tmp

If mobiliser.log shows ‘No channels found for channel id: defaultQ’ then create a ‘Preference’ for filechannel. Key=channelId Value=defaultQ see pages 12-13 of this document.

www.sap.com