

**SUSHI**

Date January 16, 2006  
Pages 2  
Subject A toolkit for accessing the sushi Web - Quick start guide

**WHAT'S IN THERE**

There are two toolkits; one of them allows one to set up an un-secure connection to the Sushi web service. The second one allows one to set up a secure connection.

Both toolkits are simple exports from Eclipse (version 3.0), so they can be imported 'as is' within an Eclipse simple java project (not tomcat). In such a case, it is important to add all jars to the java build paths (right click on the project/properties/java build path). The class TestCase.java can then be run as a java Application (it contain a main method). All result will be displayed in the console.

To run TestCase.java on command-line, one must add all jars to the classpath, as well as the current directory (.)

**HOW TO (STEP 1: UNSECURE CONNECTION)**

The toolkit contains all the java files to set up a connection out of the box. Most of the files have been auto-generated using the Axis Wsd12Java utility, so they should not be modified. The only files that can be modified are:

- **sushi\_ws/SushiWsPortLocator.java:** This java class holds the connection parameters – where the only variable value should be the address of the service. By default, the toolkit points to the Swets web service. To change the address, one must modify the line:

```
private java.lang.String SushiWsPort_address =  
"http://psea2.swetswise.com/swetsfo/services/sushi-wsPort";
```

To another value (for example the EBSCO web service address)

- **TestCase.java:** This file holds the report parameters. It looks like this:

```
Requestor requestor = new Requestor();  
requestor.setId("A-REQUESTOR-ID");  
requestor.setName("A-REQUESTOR-NAME");  
requestor.setEmail("A-REQUESTOR-EMAIL");  
rptRequest.setRequestor(requestor);  
  
CustomerReference customer = new CustomerReference();  
customer.setName("A-CUSTOMER-NAME");  
customer.setId("7763");  
rptRequest.setCustomerReference(customer);  
  
ReportDefinition reportDefinition = new ReportDefinition();  
ReportDefinitionFilters filter = new  
ReportDefinitionFilters();
```

```

Range range = new Range();
SimpleDateFormat formatter= new SimpleDateFormat ("yyyy-MM-dd");
range.setBegin(formatter.parse("2005-05-01"));
range.setEnd(formatter.parse("2005-05-31"));
filter.setUsageDateRange(range);

```

Most of those parameters will simply be reflected in the generated report, but the values in `customer.setid`, `range.setBegin` and `range.setEnd` are important since they will inform the engine behind the Sushi Web Service which account is to be reported on, and for which date range.

The name of the report (`reportDefinition.setName("Report 1 (J1)");`) should not be changed.

## HOW TO (STEP 2: SECURE CONNECTION)

To set up a secure connection one should modify the same files as from the un-secure connection (`sushi_ws/SushiWsPortLocator.java` and `TestCase.java`). Next to that the SOAP `userid/passwords` have to be set in the files:

- **client\_deploy.wsdd:** This file contains the base settings for the soap secure connection, as such it also contains the user id to be sent to the sushi engine.

```

<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <transport name="http"
pivot="java:org.apache.axis.transport.http.HTTPSender"/>
  <globalConfiguration >
    <requestFlow >
      <handler type="java:org.apache.ws.axis.security.WSDoAllSender" >
        <parameter name="action" value="UsernameToken"/>
        <parameter name="user" value="test1"/>
        <parameter name="passwordCallbackClass" value="PWCallback"/>
        <parameter name="passwordType" value="PasswordDigest"/>
      </handler>
    </requestFlow >
  </globalConfiguration >
</deployment>

```



- **PWCallback.java:** This file handles the password lookup for SOAP, it therefore contains the password for a given `userid`.

```

public void handle(Callback[] callbacks) throws IOException,
    UnsupportedCallbackException {
    for (int i = 0; i < callbacks.length; i++) {
        if (callbacks[i] instanceof WSPasswordCallback) {
            WSPasswordCallback pc = (WSPasswordCallback)callbacks[i];
            // set the password given a username

            if ("test1".equals(pc.getIdentifer())) {
                pc.setPassword("test1");
            }
            else {
                throw new UnsupportedCallbackException(callbacks[i], "Unrecognized Callback");
            }
        }
    }
}

```

For Swets, the allowed user-id passwords to be used are `test1/test1`, `test2/test2` or `test3/test3`.