

## Toolkit & Demonstrator Final Report Template (Cut-Down Version)

This template is very loosely based on the JISC Final Report Template available from the [JISC Project Management Guidelines](#) web page.

<b>Project name/acronym:</b>	SOCKET (Service-Oriented Consumer Kit for Elf Tools)
<b>Project website/blog address:</b>	<a href="http://www.socketelf.org:8080/eldg/socket/">http://www.socketelf.org:8080/eldg/socket/</a>
<b>Report, author(s):</b>	Andrew Booth
<b>Contact person</b> (if different from above):	
<b>Date:</b>	14 <sup>th</sup> September 2006

### Methodology

The project methodology involved dividing the project into roughly 5 loosely coupled sub-projects, with one person taking the lead for each: creation of the core SOCKET Java code; construction of the XSLT view transformations; Bodington, Guanxi and Perception; and the jUDDI registry.

This approach worked well and enabled us to fulfil our main objectives. The major challenges had been identified correctly before the project started. The first was to produce XSLT stylesheets that could deal with the wide range of possible combinations of input and output parameters. The second was dealing with the wide range of possible WSDL styles.

The UDDIbrowser and JAXP technologies were not used as UDDI4J proved sufficient for our purposes.

### Implementation

Not evident from the project plan is the extent of modularisation of the SOCKET codebase. The SourceForge sub-projects attest to this: socket-engine; socket-engine-client; consumer-factory; socket-view; socket-console; and QMShibb. This modularity, a strategy of coding to interfaces, and use of the factory design pattern will help future development of the project.

The heart of the SOCKET functionality is the extraction and structuring of input and output parameter metadata from the consumer software stubs produced by the Axis toolkit. This proved to be the major challenge of the project. The beta code presently handles all simple datatypes and arrays thereof. Release 1.0 will follow shortly and will be able to deal with complex datatypes.

The Rose implementation of the consumer factory uses Freemarker templates to generate the code that entails the Web service consumer software. This is bundled together with the view layer to form a Java WAR archive.

XML Schemas were developed to provide an abstract description of the user interfaces.

The view layer has been created from XSLT transformations of the abstract user interface XML carried out in a servlet filter. Custom XSLT stylesheets can be applied to each service and to each operation within a service. There is client-side validation of input data.

Modifications were carried out on the Bodington VLE so that it can incorporate SOCKET Web services and applications. The link to SOCKET services is achieved by the inclusion of a socket2bods module in the Bodington code. For simple applications this jar can contain a registry of available services. The other option is to use the code in socket2bods to communicate with an external UDDI web services registry.

The QMShibb software module was developed to provide a Guanxi/Shibboleth guard to the Questionmark Perception assessment software. Single sign-on access to all functionality is now possible from Bodington and other VLEs that have access to a Guanxi engine.

In order to coordinate the range of functions that have emerged in SOCKET - management of WSDL, Web service and registry functions – a management console Web application has been built.

## **Outputs and Results**

### **The SOCKET Engine**

The Socket Engine is a consumer factory that takes a WSDL document and produces a WAR file containing Web service consumer software and an XSLT view layer. The WSDL may contain simple data types and arrays of simple data types. We have so far generated WAR files from the WSDL documents of the following services:

ioMorphWSXcri  
ioMorphWS IMS ENT  
Questionmark QMWISE  
SSRun (from the JISC ASSIS project)

### **SOCKET Pluto**

Pluto is a command line client for the Socket Engine. Given a URL to a WSDL document, Pluto returns a URL to a download of the consumer WAR file.

### **SOCKET Saturn**

Saturn is a web application client for the Socket Engine. Given a URL to a WSDL document through an HTML form, Saturn returns a hyperlink to a download of the consumer WAR file.

### **QMShibb**

QMShibb is a web application written in Java that allows Questionmark Perception to be used with Shibboleth. It links a Shibboleth Service Provider to the QMWISE web services interface

### **SOCKET Bodington**

This is a version of the Bodington VLE which has been extended to include a tool for the creation of SOCKET links which allow external web service learning tools (or any web application) to appear as Bodington tools.

The SOCKET binaries and documentation are available from the project website  
<http://www.socketelf.org:8080/eldg/socket/>

The source code is available from SourceForge at <http://sourceforge.net/projects/socket>

The labelling and descriptions of the input fields of the SOCKET-generated user interface have yet to be finalized, but that is simply a matter of either using another XSLT transform or using a programmatic transform in a servlet filter.

## **Implications**

SOCKET provides a basis for a Service-oriented virtual learning environment (SOVLE) in which collections of learning services can be gathered together and presented to users via a customised interface. SOCKET has demonstrated the need for a registry behind the system to allow the management of what could otherwise be a tangled web of inter-relating services. We would hope that future development would include the development of other consumer factories that specialise in different type of web services.