SDN Contribution

WS-I Sample Application Blog Series: WS-I Sample Application in action

Applies to:

SAP NetWeaver Web Application Server 2004s Java Stack, SPS 7

Summary

If you are a devoted reader of this blog series, you should now have a profound understanding of the design and implementation details of SAP's new WS-I Sample Application core functionality and security configuration. In this last installment, we'll demonstrate the usage of the Sample Application and test interoperability with another vendor's Sample Application that has also implemented the WS-I Basic Security Profile Working Group Draft. This will take us through a detailed walkthrough of the user interface and we'll also have a chance to look at the advanced logging and monitoring features that are important for interoperability testing.

Created on: 7 August 2006

Author Bio

As a Standards Architect with SAP's Industry Standards team, Martin works in the area of standardization and interoperability testing of new Web Services technologies, focusing on message security and identity management. Martin is a frequent speaker at conferences and author of books and articles relating to information security, integration middleware and J2EE development.

Table of Contents

Sample Walkthrough 2

Configuration page: Starting a new demo run 2

Retailer Catalog page: Submitting the Order 5

Order Status page: Checking the results of the order 6

Track Order page: What happened behind the scenes? 6

Conclusion 7

Disclaimer and Liability Notice 7

Sample Walkthrough

To run the Sample App on the NetWeaver WebAS 2004s Java Stack, you must first complete the installation steps as described in the `readme.txt` file that is included in the download archive (http://www.ws-i.org/SampleApplications/BSP/SAP-BSP-SA-2006.05.zip). After successful installation, you can start the Sample App by invoking the URL http://localhost:53000/webdynpro/dispatcher/sap.com/test~wsi-wd/WSISampleApplication from a Web Browser (you may have to modify the hostname and port number in this URL according to the http configuration of your engine's dispatcher). The overall page flow is as follows:

- **Login page**: Use a valid user account to login to the WS-I Sample Application
- **Configuration page**: This is where all settings are configured prior to a new demo run. This includes selection of the service endpoints for each role defined in the SCM scenario as well as proxy settings for monitoring or firewall reasons (more on this in a minute)
- **Retailer Catalog page**: The Retailer's website for consumers who want to purchase electronic products from the Retailer's catalog. The consumer has to enter the quantities for each product and can then submit the order to the Retailer System.
- **Order Status page**: Once the order has been validated by the Retailer System, the status of the order is displayed which includes the number of line items that have been shipped and the line items that could no be shipped.
- **Track Order page**: During the course of each demo run, a series of log events are captured behind the scenes by the Logging Facility and are displayed on this page to track the message flow of the order. The "Back to Configuration Page" link on this page starts a new demo run.

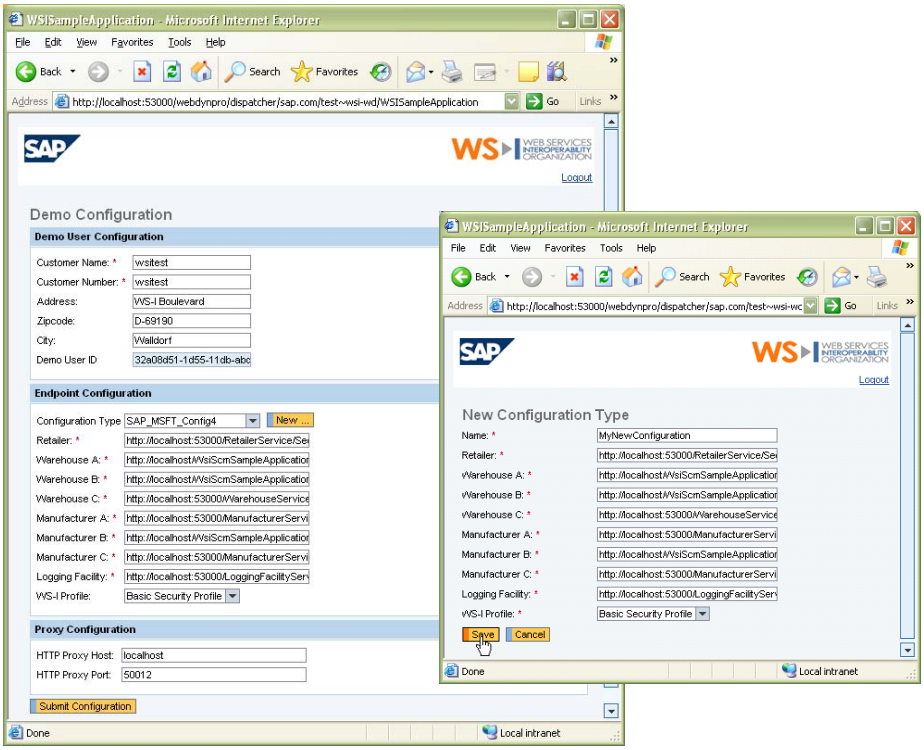
Configuration page: Starting a new demo run

After successful login, the Web Client requests the Demo Configuration page. This page is divided into three subsections:

- **Demo User Configuration**: This data is retrieved from the User Management Engine (UME) in WebAS and used for the customer details in some of the WS-I SCM message headers. The Demo User ID is an auto generated unique id which is used to track and retrieve the log entries for different (concurrent) users. It is similar to a session id which is valid for one demo run.
- **Endpoint Configuration**: There are preconfigured values for the endpoint URLs (e.g. local SAP service URLs for both BP and BSP) available by choosing one of the entries from the Configuration Type drop down list. It is also possible to override these default values or create additional

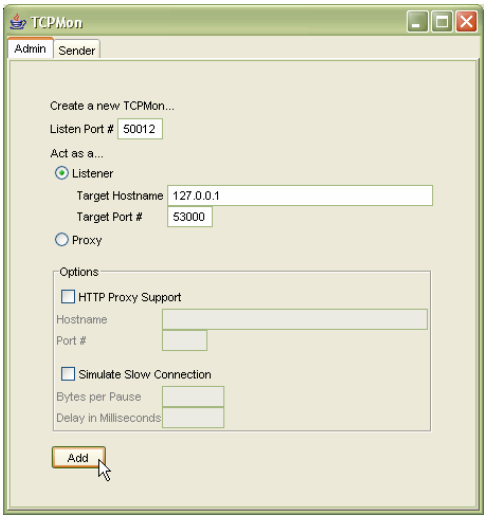
configuration types by clicking on the “New…” button. A configuration type always consists of a unique name, the URLs assigned to the different actors in the SCM scenario and the supported WS-I profile of these endpoints. At the moment, BP 1.0 and BSP 1.0 are supported by SAP’s Sample App.

- **Proxy Configuration**: Web Service interoperability testing is always associated with detailed analysis of the exchanged messages between a consumer and provider. Therefore, it is helpful to have a message tracing facility which can intercept the communication and log the data for later use. Furthermore, the Web Service endpoints of another vendor’s implementation might be behind a firewall and can’t be invoked directly. In both cases, SAP’s Sample Application supports the configuration of an HTTP proxy. Once a hostname and port number is entered, each call initiated from a Web Service client is first directed to this proxy and then forwarded to the final destination.



The following test scenario illustrates how to utilize this feature in order to trace the messages exchanged between local instances of SAP’s and Microsoft’s BSP Sample Application implementations. That is, both applications are deployed on the same host. To download another vendor’s implementation, simply check for its availability at the WS-I Sample Applications Working Group home page (http://www.ws-i.org/deliverables/workinggroup.aspx?wg=sampleapps) on the WS-I Web site.

To capture the SOAP messages, I recommend the Apache TCPMon tool which was originally part of the Axis 1.x Web Services framework and is now an independent project, available for download at http://ws.apache.org/commons/tcpmon/. The installation procedure is very simple: Unzip the archive to a directory and start the batch file tcpmon.bat (tcpmon.sh on UNIX) from the /build subdirectory.



TCPMon comes up with the Admin dialog where one can start new message monitors, each listening on a distinct TCP port. Pick a listen port that is not in use by any other application (e.g. 50012) and configure the new monitor to act as a listener, forwarding any requests/responses to the target host name of your WebAS instance where the WS-I Sample Application is running and the HTTP port configured for this instance (e.g. 53000). After clicking on the 'Add' button, the new monitor is waiting for messages. I recommend activating the 'XML Format' checkbox at the bottom of the monitor dialog to increase readability of the traces.

Back to the Demo Configuration screen in the WS-I Sample Application: In order to make the WS-I Sample Application aware of the message monitor, the port number and host name (or IP address) where the monitor listens must be entered in the Proxy Configuration fields. After clicking on the "Submit Configuration" button, the first message (getCatalog) is sent to the endpoint configured for the Retailer and both request and response are captured by the monitor.

In the sample message trace above, a local deployment of Microsoft's WS-I BSP Sample Application implementation (also available from the WS-I Sample Application Working Group page at http://www.ws-i.org/deliverables/workinggroup.aspx?wg=sampleapps) was used to capture SAP's Web Client getCatalog invocation against another vendor's Retailer System implementation. This trace allows you to have an in-depth look at the message exchange and even let you resubmit the same request again from the monitor.

Setting an HTTP proxy at runtime for a Web Service invocation of a deployable proxy can be achieved by configuring the respective properties of the logical port used to call the service provider (code excerpt from the RetailerSystemBean orderFromWarehouse method):

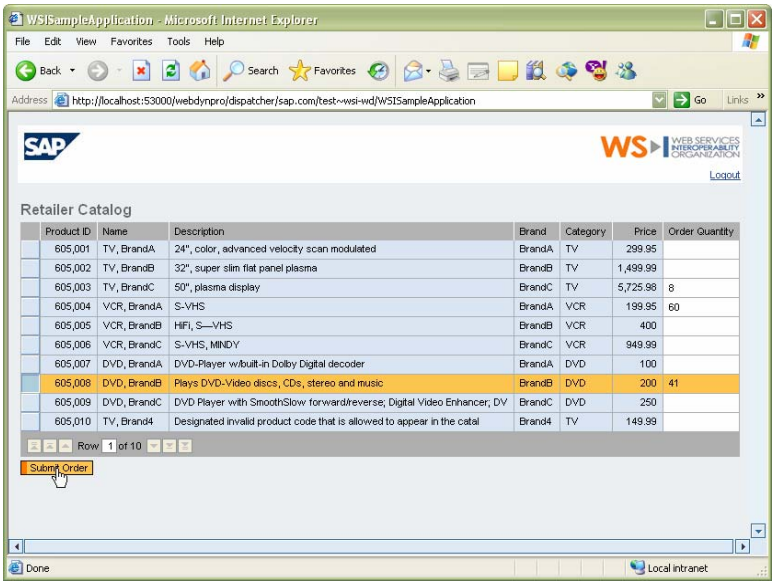
```
warehouse._setProperty("javax.xml.rpc.http.proxyhost",
   DemoConfiguration.getInstance().getHttpProxyHostForDemo(demoUserId));

warehouse._setProperty("javax.xml.rpc.http.proxyport",
   DemoConfiguration.getInstance().getHttpProxyPortForDemo(demoUserId));
```

DemoConfiguration is a singleton that manages all demo configurations as part of the Demo System functionality. Each configuration (endpoints, proxy name and port) is assigned to the unique demo user id.

Retailer Catalog page: Submitting the Order

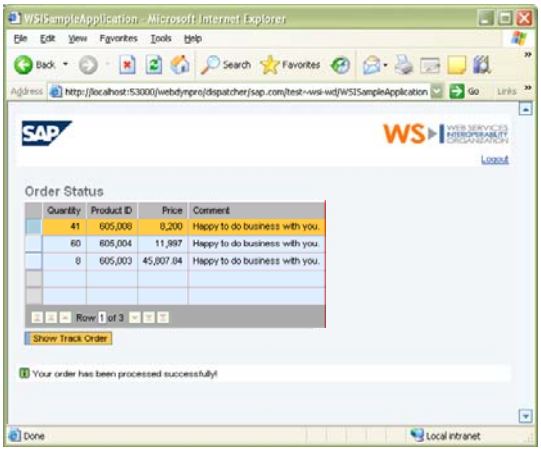
Once a configuration is submitted by the user, the actual SCM scenario starts and the Retailer System responds with the content of the catalog. From the list of products on the Retailer Catalog page, the consumer can enter the order quantity of each product and submit the order to the Retailer System.



Now the order is validated. This means that the Retailer System determines which Warehouse can supply each item and asks the Warehouse(s) to ship them. If the inventory level for a product in a warehouse falls below its minimum level, the Warehouse purchases goods from a Manufacturer to replenish stock for the particular product. The Manufacturer ships the goods and sends a shipping notice to the Warehouse. In contrast to all other synchronous message exchanges, the communication between the Warehouse and the Manufacturer is based on an asynchronous callback. This is because the manufacturer cannot always respond immediately to the purchase order from the warehouse. It may already have the goods or may have to schedule a production run.

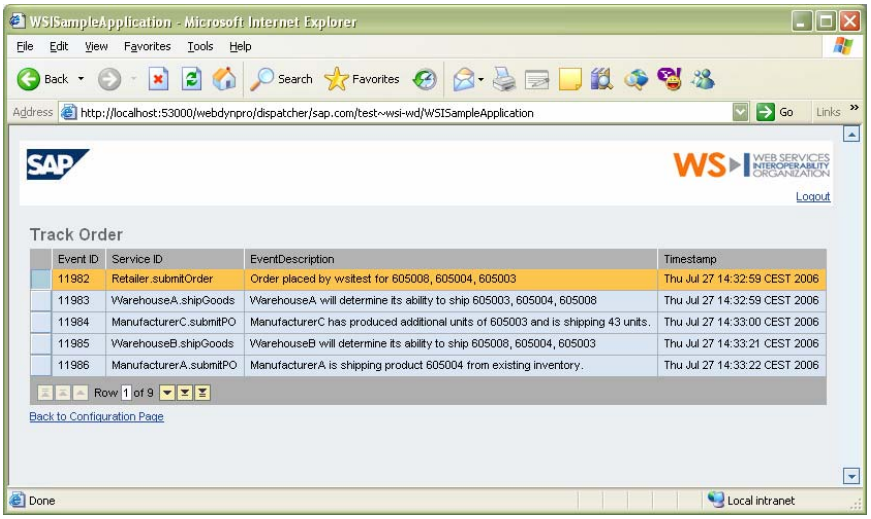
Order Status page: Checking the results of the order

Once the Retailer System has the ShipGoods responses from all warehouses, it returns the submitOrder response message to the Web Client with a status of each order, along with its price and whether the item order can be fulfilled.



Track Order page: What happened behind the scenes?

Finally, the user can see the captured log events on the Track Order page. This log reveals which Web Services have been consumed by a given operation and the outcomes of those Web Services. In addition, the corresponding messages can be viewed in the TCPMon monitor (if configured).



Conclusion

We hope you find this blog series of value in your own projects where interoperability between different Web Service platforms is a key success factor. The WS-I Sample Application is a not only a means to test interoperability between the vendors that provide implementations of WS-I profiles. They are also a reference implementation for the BP and BSP support in SAP NetWeaver WebAS by illustrating how to build resilient, secure and interoperable Web Services based on a real-world Supply Chain Management scenario. In addition, the Sample App demonstrates the application of architectural principles such as a clear separation of cross-cutting concerns like logging, data access and exception handling in the context of J2EE development on the NetWeaver platform.

Disclaimer and Liability Notice