Apache Stonehenge : Stonehenge Interoperability Walk-through

This page last changed on May 08, 2009 by bendewey.

Overview

This document is meant to serve as a guide for exploring the interoperability of the Apache Stonehenge project. This document will walk the developer through the process of configuring the different modules to consume business services and order processing services on different frameworks.

This document assumes an environment based on a single Windows 2008 Server running all the components of the Stonehenge project. In a real world environment this application would most likely be distributed across multiple servers. Throughout this document we will refer to the services using their localhost:{port number} addresses; if your configuration is distributed over several machines, or differs otherwise, please change the localhost designation to the proper machine/host name.

Prerequisites

In order to perform the tasks in this guide, you will need to have the different components installed. Please make sure that you completed the guides on the StoneHenge wiki.

- Stonehenge .NET StockTrader Installation Guide
- Stonehenge PHP and WSAS Stocktrader Installation Guide

Service Configuration Overview

.NET Configuration

The .NET Services are managed through the web and service configuration files. In the root directory of the StockTrader client website there is a web.config file. In the appSettings section you will see an ACCESS_MODE setting, the following table shows a list of available options for the ACCESS_MODE setting:

ACCESS_MODE	Description
InProcess	Communicates directly with the database. In this access mode no external services are invoked.
DotNet_Http_WcfService	Communicates with the .NET Business WCF Service using BasicHttpBinding at http://localhost:9000/TradeBusinessService
DotNet_WsHttp_WcfService	Communicates with the .NET Business WCF Service using WsHttpBinding and Secure Messaging at

	http://localhost:9000/TradeBusinessService/msec
PHP_Http_WebService	Communicates with the PHP Web Service using http at http://localhost:8080/php_stocktrader/business_service/busines
WSAS_Http_WebService	Communicates with the WSAS Web Service using http at http://localhost:9763/services/TradeServiceWsas

After a buy/sell operation is sent to the Business Service it will be routed to the OrderProcessing Service. The OrderProcessingService is configured using the Trade.BusinessServiceConsole.exe.config file in the business_service\bin\Debug directory. In the appSetting section you will see a setting for ORDER_PROCESSING_MODE. Below is a table that describes the different options for the order processing mode. With the exception of the InProcess mode, the order is processed asynchronously, the call will return immediately and the user will see the Closed Order Alert after visiting a subsequent page.

Note:
If you change the order processing mode you will have to restart the console application for the changes to take effect. This is not the case for the .NET StockTrader website, any changes to the config will be reloaded on the next page request.

ORDER_PROCESSING_MODE	Description
Sync_InProcess	Communicates directly with the database. In this order processing mode the business service will process the order synchronously and will return with the order has processed
ASync_DotNet_Http	Communicates with the .NET Order Processing WCF Service using BasicHttpBinding. The default order processor address for this mode is http://localhost:8000/tradeorderprocessor
ASync_DotNet_WsHttp_MSecurity	Communicates with the .NET Order Processing WCF Service using WsHttpBinding and Secure Messaging. The default order processor address for this mode is http://localhost:8000/tradeorderprocessor/msec
ASync_PHP_Http	Communicates with the PHP Order Processing Service using http. The default order processor address for this mode is http://localhost:8080/php_stocktrader/order_processor/order_p
ASync_PHP_WsHttp_MSecurity	Communicates with the PHP Order Processing Service using WsHttp and Secure Messaging. The default order processor address for this mode is http://localhost:8080/php_stocktrader/order_processor/order_p
ASync_WSAS_Http	Communicates with the WSAS Order Processing Service using http. The default order processor address for this mode is http://localhost:9763/services/OrderProcessor
ASync_WSAS_WsHttp_MSecurity	Communicates with the WSAS Order Processing Service using WsHttp and Secure Messaging. The default order processor address for this mode is

	http://localhost:9763/services/OrderProcessorMsec

PHP/WSAS Configuration

The PHP and WSAS Services are managed through the StockTradeDB database. After you run the PHP database scripts you will have four new tables (DbConfig, ClientToBs, BsToOps, and Service), which are used to configure the services for PHP/WSAS.

Before you change the ClientToBs and BsToOps configuration open the Service tables and verify that the endpoint addresses are accurate for your environment. For this walk-through you will need to have the following endpoints.

Service Name	Default Value
DOTNET_BS	http://localhost:9000/TradeBusinessService
DOTNET_OPS	http://localhost:8000/TradeOrderProcessor
DOTNET_OPSSEC	http://localhost:8000/TradeOrderProcessor/sec
JAVA_BS	http://localhost:9763/services/TradeServiceWsas
JAVA_OPS	http://localhost:9763/services/OrderProcessor
JAVA_OPSSEC	http://localhost:9763/services/OrderProcessorMsec
PHP_BS	http://localhost:8080/php_stocktrader/business_service/busines
PHP_OPS	http://localhost:8080/php_stocktrader/order_processor/order_p
PHP_OPSSEC	http://localhost:8080/php_stocktrader/order_processor/order_p

Note:
In the above service names BS stands for Business Service, OPS stands for Order Processing Service, and OPSSEC stands for Order Processing Service with Secure messaging. Throughout the remainder of this document the abbreviations BS and OPS will refer to the Business Service and Order Processing Service respectively. OPS will be used to refer to OPS or OPSSEC depending on your configuration.

Note 2:
In this edition of the interop guide WSAS is the only JAVA implementation and the two terms may be used interchangeably.

To configure the PHP Client

1. Open the ClientToBS table
2. Locate the PHP_CLIENT row
3. Change the BS value to the corresponding BS service name from the table above.

To configure the PHP Business Service

1. Open the BsToOps Table
2. Locate the PHP_BS row
3. Change the OPS value to the corresponding OPS or OPSSEC service name from above.

To configure the WSAS Business Service

1. Open the BsToOps Table
2. Locate the JAVA_BS row
3. Change the OPS value to the corresponding OPS or OPSSEC service name from above.

In addition to the Business Service and the Order Processing Service the PHP StockTrader website has a Configuration Service. This configuration service is used to pull the data from the database and determine which services to use in real-time. To setup the Configuration Service you will need to ensure that the resources\conf\database_config.xml file has the proper database connection information. Additionally, you will need to set the configuration service endpoint address under the Config tab of the PHP website. The default is http://localhost:8080/php_stocktrader/config_service/config_svc.php. For more information on setting up the PHP configuration see the install guide in the Prerequisites section.

Interop Walk-through

Now that you are familiar with the configuration for the different services we are going to walk through the different configurations. Since all the services are communicating with the same database you will be able place an order in the .NET StockTrader website then sell the same order from the PHP website.

Getting Started

In order to get started you should open and run all the services. Please follow these steps to start all the services and websites.

1. Start the .NET Services
    a. %dotnet_root%\RunServices.bat (be sure to Run as Administrator)
2. Start the WSAS server
    a. Windows: %wsas_server_root%\bin\wso2server.bat (be sure to Run as Administrator)
    b. Linux/Unix: %wsas_server_root%\bin\wso2server.sh
3. Open the .NET StockTrader website
    a. http://localhost/trade
4. Open the PHP StockTrader website
    a. http://localhost:8080/php_stocktrader/trader_client/
5. Open SQL Management Studio and connect to the StockTraderDB.

Warning
If you have trouble running any of these services or websites please refer back to the installation guides in the Prerequisite section before continuing.

Placing an order on the .NET Stack

1. From the http://localhost/trade website click Login from the top menu.
2. Enter the StockTrader credentials of:
    a. username = uid:0
    b. password = xxx

3. Click Login. This will authenticate you using the BS.
4. Click Quotes/Trade from the top menu. This will pull a list of available stocks from the BS
5. Click the Buy link next to one of the stocks
6. Enter 111 for the number of shares and click Buy
7. Under the default configuration the .NET website does the following
    a. Sends a message to the BS to buy a stock.
    b. The BS creates a record in the database for the order with a status of open
    c. The BS then sends an asynchronous call to the OPS.
    d. At which point the page is returned with a message stating that the trade has been submitted for processing.
8. In the background the Order Processing Service performs its processing of the order, when it has completed it updates the order in the database to closed. The closed orders will be shown to the customer the next time they request a page from the StockTrader website. Lets view the closed orders from our PHP StockTrader website

Viewing a closed order from PHP Stack

1. Open the http://localhost:8080/php_stocktrader/trader_client/ page
2. Click the Login button on the top menu
3. Enter the StockTrader credentials of:
    a. username = uid:0
    b. password = xxx
4. When you login the PHP website will authenticate you using the BS.
5. After you have logged in you will be taken to the Accounts page, where you should be greeted with a Trade Alert, stating that your order placed in the .NET StrockTrader website has completed.

Trade Alert: The following orders have completed.

Order ID	Order Status	Creation Date	Completion Date	Txn Fee	Type	Symbol	Quantity
100000106	closed	04/23/2009 03:49:00 PM	04/23/2009 03:50:00 PM	$15.95	buy	s:0	11

Creating an order with .NET using the PHP BS and OPS

1. Open the .NET web.config from your Trade root folder
2. Locate the ACCESS_MODE appSetting
3. Change the value to PHP_Http_WebService
4. Save and Close the web.config
5. Switch back to the .NET StockTrader website
6. Click the Quotes/Trade tab from the top menu.
7. Click buy on one of the trades
8. Enter a quantity of 112 for the number of shares and Click Buy
9. You should now receive a submitted for processing message.
10. Click the Account tab on the top menu
11. If the PHP OPS has processed the order you should receive a Trade Alert.

Creating an order with PHP using the JAVA BS and OPS

1. Open SQL Management studio.
2. Connect to your StockTraderDB database.
3. Open the ClientToBs table for editing (Right click -> Edit Top 200)
4. Change the BS value for PHP_CLIENT to JAVA_BS
5. Open the BsToOps table for edit
6. Verify that the JAVA_BS is using the JAVA_OPS
7. Click back to the PHP website
8. Click the Quotes/Trade tab from the top menu
9. Choose a stock and click Buy
10. Enter a quantity of 113 for the number for shares
11. Click Buy
12. After the order has been submitted for processing click the Accounts tab to view the Accounts page
13. If the WSAS BS processed your order you should receive a Trade Alert.

Selling a Trade with PHP using the .NET BS and OPS

1. Again, without closing the PHP website, open SQL Management Studio
2. Open the ClientToBs table
3. This time edit the BS value of the PHP_CLIENT to DOTNET_BS
4. You don't need to change anything in the BsToOps, the .NET BS will use whatever Order Processing Mode is setup in its configuration, which should be .NET OPS.
5. Click back to the PHP website
6. Click the Portfolio tab from the top menu
7. Find the order that we created with a quantity of 111 and choose Sell
8. On the confirmation page click Sell

Selling a Trade with .NET using PHP BS and .NET OPS

1. Open SQL Management Studio
2. Open the BsToOps table
3. Change the OPS value for PHP_BS to DOTNET_OPS
4. Switch back to the .NET Website
5. Click the Portfolio tab from the top menu
6. Find the order that we created with a quantity of 112 and choose Sell
7. On the confirmation page click Sell

Selling a Trade with .NET using PHP BS and JAVA OPSSEC

1. Open SQL Management Studio
2. Open the BsToOps table
3. Change the OPS value for PHP_BS to JAVA_OPSSEC
4. Switch to the .NET StockTrader website
5. Click the Portfolio tab from the top menu
6. Find the order that we created with a quantity of 113 and choose Sell
7. On the confirmation page click Sell
8. This order has just gone from the .NET website, to the PHP BS webservice, which updated the database with the order, then using an encrypted message channel sent the order to the WSAS OPS to process the order. After the order was processed the WSAS service updated the order in the database to closed.

9. If you click the Accounts page, the .NET website will call back to the PHP BS and check for any closed orders, if any are found you should receive a trade alert.

Selling a Trade with PHP using the .NET BS and JAVA OPSSEC

1. Open SQL Management Studio
2. Open the ClientToBs table
3. Ensure that the PHP_CLIENT is configured to use the DOTNET_BS
4. Open business_process/bin/{debug|release}/Trade.BusinessServiceConsole.exe.config
5. Change the ORDER_PROCESSING_MODE under appSettings to ASync_WSAS_WsHttp_MSecurity.
    a. You may want to enable the logging mode of the WSAS Web Service. See the PHP and WSAS install documentation for logging options.
6. Save and close the config file
7. Open your .NET StockTrader Business Services Host
8. Hit Ctrl+R to restart the service.
9. Open the PHP Website
10. Click the Portfolio tab from the top menu
11. Click Sell on the first order.
12. Confirm the Sale
13. This order will be processed by the .NET Business service, which we just restarted on the fly to use the JAVA Order Processor. Additionally the order was processed with an encrypted SOAP message .

Conclusion

The Stonehenge application provides, and this paper has documented, a full end-to-end service oriented application that allows for interoperability between different platforms. With the implementation of the WS-* Standards, developers get the benefit of distributed applications and platforms, while retaining reliable and secure messages between systems. This opens the door for technologies to co-exist in today's enterprise environments.

The Stonehenge project hopes to continue to develop example applications that use the standards currently defined by the W3C and OASIS protocols. As always, the Apache Foundation encourages contributions. For more information about the project, please see our proposal at http://wiki.apache.org/incubator/StonehengeProposal.