.NET WCF Flowed Transactions

This topic covers using .NET WCF flowed transactions with SOAtest.

Sections include:

SOAtest and .NET WCF Flowed Transactions

SOAtest can initiate flowed transactions using WS-Atomic Transaction or the Microsoft OleTransactions protocol. Flowed transactions can be used when the service endpoint's binding configuration has `transactionFlow="true" and either transactionProtocol="OleTransactions"` or `transactionProtocol="WSAtomicTransactionOctober2004"`.

If the operation contract for the Web service uses the `TransactionFlow` attribute with the `TransactionFlowOption.Allowed` property, then enabling transactions in SOAtest is optional and does not require any extra test configuration.

However, if the operation contract is using the `TransactionFlowOption.Mandatory` property, then transactions are required and the test suite must be configured to initiate a flowed transaction. For general information about WCF flowed transactions, see the following:

- **WS Transaction Flow:** http://msdn.microsoft.com/en-us/library/ms752261.aspx
- **TransactionFlowAttribute**: http://msdn.microsoft.com/en-us/library/system.servicemodel.transactionflowattribute.aspx
- **TransactionFlowOption:** http://msdn.microsoft.com/en-us/library/system.servicemodel.transactionflowoption.aspx

The above WS Transaction Flow MSDN article explains what is required to configure a WCF web service client to initiate a flowed transaction. The steps necessary to configure transactions in SOAtest are similar or analogous to what is described in this article. It is highly recommended for users to read the WS Transaction Flow article in order to first understand the basic concepts behind transactions in WCF.

Configuring MSDTC

The Microsoft Distributed Transaction Coordinator (MSDTC) must be configured correctly on the machine running SOAtest, as well as the machine hosting the Web service. The MSDN WS Transaction Flow article has instructions for configuring MSDTC and how to configure the Windows firewall to allow DTC. Here are a few other articles that may be useful when configuring MSDTC.

- **How to enable network DTC access in Windows Server 2003**: http://support.microsoft.com/kb/817064/en-us
- **How to troubleshoot MS DTC firewall issues**: http://support.microsoft.com/kb/306843/en-us

Configuring the Test Suite

Invoking several Web service operations within the scope of a transaction requires configuring a `TransactionScope` object. Take the following C# code snippet:

```
CalculatorClient client = new CalculatorClient
("transactionEnabledEndpoint");using (TransactionScope tx = new
TransactionScope(TransactionScopeOption.Required)) {
        client.add(1, 2);
        client.add(2, 3);
        tx.Complete();
}
client.Close();
```

In SOAtest, an equivalent test suite would be structured as follows:

- Extension Tool 1: Create Transaction Scope
    - XML Data Bank: propagation token

- Tool 1: add(1,2)
- Tool 2: add(2,3)
- Extension Tool 2: Complete Transaction

In the test suite above, Extension Tool 1 is needed to create the `TransactionScope` object and to get the propagation token for the current transaction. The propagation token is then put into an XML Data Bank because it will be needed later by the tools (described later). Here is sample Jython code that can be used for Extension Tool 1:

```
from webtool.soap.wcf import *;
from soaptest.api import *;

def createTransaction(input, context):
   id = WcfUtil.createTransactionScope(WcfUtil.
TRANSACTION_SCOPE_OPTION_REQUIRED, WcfUtil.ISOLATION_LEVEL_SERIALIZABLE,
60000)
   context.put("MY_TRANSACTION_ID", id)
   token = WcfUtil.getCurrentPropagationToken()
   return SOAPUtil.getXMLFromString([token])
```

The Extension Tool 2 in the test suite is used to complete the transaction. This Extension tool is doing the equivalent of tx.Complete() in the above C# code snippet. Here is the Jython code for Extension Tool 2:

```
from webtool.soap.wcf import *;
def endTransaction(input, context):
   id = context.get("MY_TRANSACTION_ID")
   WcfUtil.completeTransactionScope(id)
```

The WcfUtil methods and corresponding .NET API are described below:

int webtool.soap.wcf.WcfUtil.createTransactionScope(int scopeOption, int isolationLevel, int scopeTimeout)

Parameters

`scopeOption`

- See TransactionScopeOption Enumeration: http://msdn.microsoft.com/en-us/library/system.transactions.transactionscopeoption.aspx
- Can be one of the following:
    - WcfUtil.TRANSACTION_SCOPE_OPTION_REQUIRED
    - WcfUtil.TRANSACTION_SCOPE_OPTION_REQUIRES_NEW
    - WcfUtil.TRANSACTION_SCOPE_OPTION_SUPPRESS

`isolationLevel`

- See IsolationLevel Enumeration: http://msdn.microsoft.com/en-us/library/system.transactions.isolationlevel.aspx
- Can be one of the following:
    - WcfUtil.ISOLATION_LEVEL_SERIALIZABLE
    - WcfUtil.ISOLATION_LEVEL_REPEATABLE_READ
    - WcfUtil.ISOLATION_LEVEL_READ_COMMITTED
    - WcfUtil.ISOLATION_LEVEL_READ_UNCOMMITTED
    - WcfUtil.ISOLATION_LEVEL_SNAPSHOT
    - WcfUtil.ISOLATION_LEVEL_CHAOS
    - WcfUtil.ISOLATION_LEVEL_UNSPECIFIED

`scopeTimeout`

- The timeout of the TransactionScope in ms.

Returns

An ID that can be used to reference the TransactionScope.

String WcfUtil.getCurrentPropagationToken()

Returns

The Base64 encoding of the propagation token for the current transaction.

WcfUtil.completeTransactionScope(int transactionScopeRef)

Parameters

`transactionScopeRef`

- The id returned by WcfUtil.createTransactionScope()

Configuring Transaction Headers

The tools each need to have a SOAP header that contains the propagation token. This propagation token comes from the XML Data Bank chained to the first Extension Tool test that was described earlier. The type of SOAP header that is needed depends on the transaction protocol.

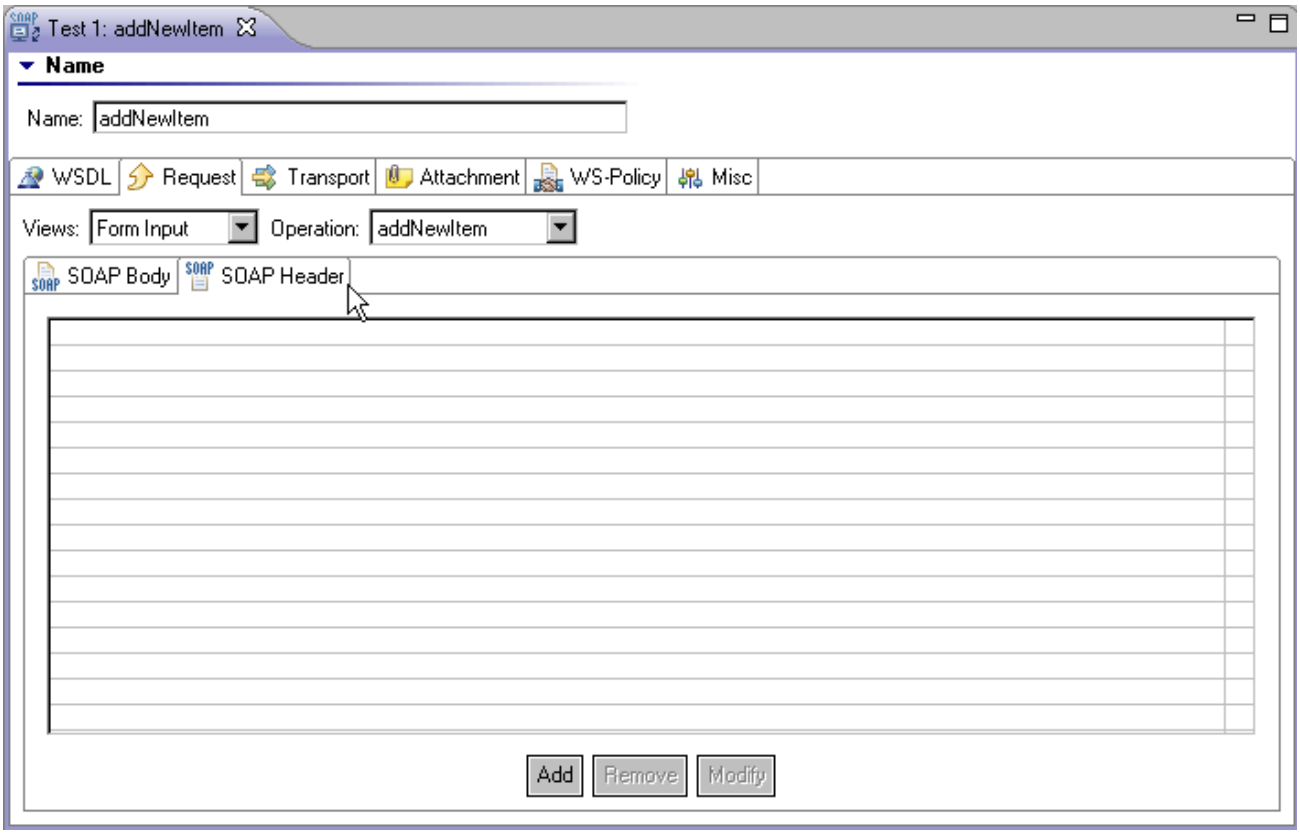
OleTransaction Headers

For OleTransactions, a OleTxTransaction SOAP header is needed. For convenience, here is a sample `OleTxTransaction` SOAP header:

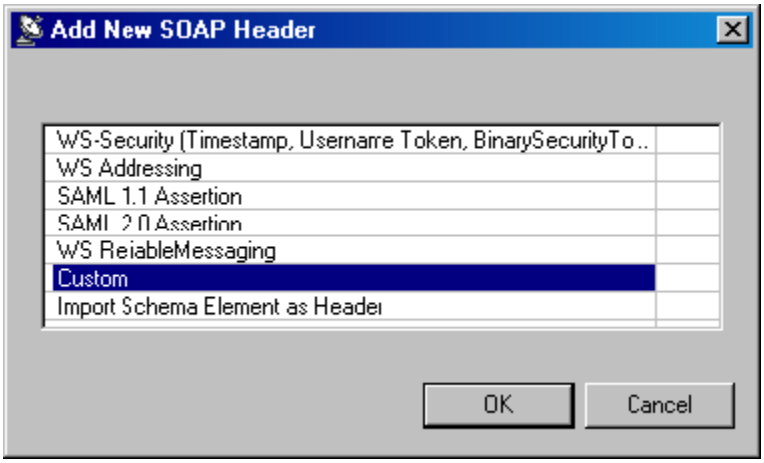
```
<OleTxTransaction b:Expires="59904" s:mustUnderstand="1"
   xmlns="http://schemas.microsoft.com/ws/2006/02/tx/oletx"
   xmlns:b="http://schemas.xmlsoap.org/ws/2004/10/wscoor"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <PropagationToken></PropagationToken>
</OleTxTransaction>
```

To add the above header to tool 1, complete the following:

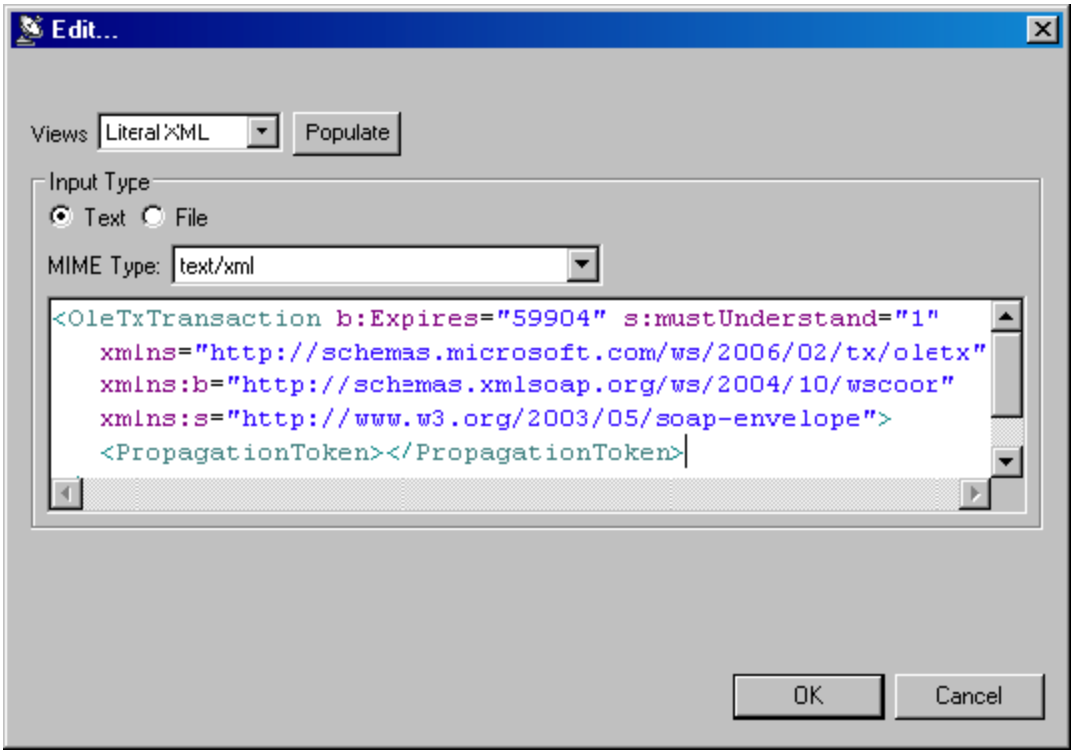
1. Select the **SOAP Header** tab from within a tool's **Request** tab.



2. Click the **Add** button within the **SOAP Header** tab. The **Add New SOAP Header** dialog displays.



3. Select **Custom** from the **Add New SOAP Header** dialog and click the **OK** button. A new row is added to the SOAP Header tab.
4. Double-click the new row in the SOAP Header tab. An **Edit** dialog displays.



5. Paste the following sample `OleTxTransaction` SOAP header into the Literal/XML text field of the **Edit** dialog:

```
<OleTxTransaction b:Expires="59904" s:mustUnderstand="1"
   xmlns="http://schemas.microsoft.com/ws/2006/02/tx/oletx"
   xmlns:b="http://schemas.xmlsoap.org/ws/2004/10/wscoor"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <PropagationToken></PropagationToken>
</OleTxTransaction>
```

6. Select **Form XML** from the **Views** drop down box of the **Edit** dialog, and click **Yes** to override Form XML values with Literal XML values.
7. In the Form XML view, select the **PropagationToken** element, then select the **Parameterized** option for the element's value. This will allow you to choose the XML Data Bank column containing the propagation token.
8. Click the **OK** button.

WS-Atomic Transaction Headers

For WS-Atomic Transaction, a `CoordinationContext` SOAP header is needed instead. Similar to the `OleTxTransaction` header, the `PropagationToken` element must be parameterized to include the propagation token from the XML Data Bank. For convenience, here is a sample CoordinationContext SOAP header:

```
<CoordinationContext s:mustUnderstand="1"
   xmlns="http://schemas.xmlsoap.org/ws/2004/10/wscoor"
   xmlns:a="http://www.w3.org/2005/08/addressing"
   xmlns:mstx="http://schemas.microsoft.com/ws/2006/02/transactions"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <wscoor:Identifier
xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor"></wscoor:
Identifier>
   <Expires>59904</Expires>
   <CoordinationType>http://schemas.xmlsoap.org/ws/2004/10/wsat<
/CoordinationType>
   <RegistrationService>
        <Address
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">https://hostname
/WsatService/Registration/Coordinator/Disabled/</Address>
        <ReferenceParameters xmlns="http://schemas.xmlsoap.org/ws/2004/08
/addressing">
               <mstx:RegisterInfo>
                      <mstx:LocalTransactionId></mstx:LocalTransactionId>
                      </mstx:RegisterInfo>
               </ReferenceParameters>
   </RegistrationService>
   <mstx:IsolationLevel>0</mstx:IsolationLevel>
   <mstx:LocalTransactionId></mstx:LocalTransactionId>
   <PropagationToken xmlns="http://schemas.microsoft.com/ws/2006/02/tx
/oletx"></PropagationToken>
</CoordinationContext>
```

To add the above header to tool 2, follow the same procedure as outlined for tool 1, but instead paste the above CoordinationContext SOAP header into the Edit dialog.